ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем

(повна назва кафедри (предметної, циклової комісії))

Кваліфікаційна робота

на здобуття рівня вищої освіти «бакалавр»

(рівень вищої освіти)

Апаратно-програмний комплекс моніторингу температурного режиму зерносховищ

(тема кваліфікаційної роботи українською мовою)

Hardware and software complex for monitoring the temperature regime of grain storage facilities

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач ВО денної форми навчання
спеціальності 123 – Комп'ютерна інженерія

(шифр і назва напряму підготовки, спеціальності)

Освітня програма «Комп'ютерна інженерія»

(назва освітньої програми)

Круш Артур Ігорович

(прізвище, ім'я, по-батькові)

Керівник д-р техн. наук, проф., Малахов Є.В

(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент Розновець О.І.

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№___ від «___»_____ 2025 р.

Завідувач кафедри
_____ Євгеній МАЛАХОВ
(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № __5__
протокол № __ від «___» ___ 2025 р.
Оцінка _____/_____/_____
(за національною шкалою, шкалою ECTS, бали)

Голова ЕК
_____ Світлана АНТОЩУК
(підпис) (ім'я, прізвище)

Одеса - 2025

АНОТАЦІЯ

Системи термометрії зерносховищ, що надають оператору підприємства можливість наочного моніторингу за температурним станом, є невіддільною частиною функціонування елеваторів. Водночас існує тенденція підвищення ефективності апаратно-програмних комплексів моніторингу температурного режиму зерносховищ та зниження витрат на модернізацію старих комплексів.

Метою цієї роботи є підвищення ефективності апаратно-програмного комплексу моніторингу температурного режиму зерносховищ, який вже впроваджено на цілому ряді елеваторів, шляхом розробки його додаткових елементів. Серед розроблених елементів комплексу:

– програмне забезпечення (прошивка) для мікроконтролера ATtiny44 написане мовою Асемблера, що реалізує як протокол обміну даними по шині 1-Wire, так і інтерфейс для емуляції взаємодії з цифровими датчиками DS18B20, з ціллю адаптації термопідвісок з аналоговими датчиками до комплексу;

– розширені компоненти візуалізації для ПЗ Viewer, що представляє показники температури у різних точках зерносховища в зручному для оператора вигляді.

Розширення візуалізації для ПЗ Viewer, яке розроблено, підвищує ефективність моніторингу, а модуль на базі контролера ATtiny44 дозволяє знизити вартість модернізації усього комплексу завдяки підтримці наявних на підприємствах старих термопідвісок із аналоговими датчиками температури.

ANNOTATION

Granary thermometry systems, which provide the operator of the enterprise with the opportunity to visually monitor the temperature state, are an integral part of the functioning of elevators. At the same time, there is a tendency to increase the efficiency of hardware and software complexes for monitoring the temperature regime of granaries and reduce the costs of modernizing old complexes.

The purpose of this work is to increase the efficiency of the hardware and software complex for monitoring, which has already been implemented on a number of elevators, by developing its additional elements. Among the developed elements of the complex:

– software (firmware) for the ATtiny44 microcontroller was developed in the Assembler language, which implements both the 1-Wire bus data exchange protocol and an interface for emulating interaction with DS18B20 digital sensors, in order to adapt bobs with analog sensors to the complex;

– extended visualization components for the Viewer software, which represents temperatures at different points of the granary in a convenient form for the operator.

The visualization extension for the Viewer software, which has been developed, increases the efficiency of monitoring, and the module based on the ATtiny44 controller allows to reduce the cost of modernizing the entire complex by supporting old bobs with analog temperature sensors, which are available at enterprises.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

ATtiny44 – Atmega tiny44

CRC – cyclic redundancy check

EEPROM – electrically erasable programmable read-only memory (енергонезалежна пам’ять, що очищується та програмується за допомогою електрики)

imm – immediate (числова константа)

MIPS – million instructions per second

Registrator – STRegistrator2000v105.exe

RISC – reduced instruction set computing (обчислення зі скороченим набором команд)

SPI – serial peripheral interface (послідовний периферійний інтерфейс)

Viewer – STViewer2000v205.exe

АЛП – арифметико-логічний пристрій

КМОН – комплементарна структура метал-оксид-напівпровідник

МК – мікроконтролер

ОЗП – оперативний запам'ятовуючий пристрій

ОМК – однокристальні мікроконтролери

ПЛК – програмований логічний контролер

РЗП – регістри загального призначення

ПЗП – постійний запам’ятовуючий пристрій

ВСТУП

Автоматизація вимірювань із забезпеченням їх моніторингу залишається трендом розвитку для промислових підприємств, зокрема для сільськогосподарської обробки. Процеси зберігання та сушіння агропродукції у силосах – спеціалізованих сховищах у складі елеватора, вимагають регулювання температури у різних точках по ширині та висоті. З цією ціллю використовуються термопідвіски, які підключаються до контролерів, і являють собою кабель-трос зі збіркою датчиків. Зі свого боку контролери передають показники на вищий рівень – до комп'ютера оператора. Актуальною проблемою є розробка елементів для вже існуючих систем, які б зробили моніторинг температури більш докладним, ефективним та зручним для персоналу.

Одним з найефективніших засобів забезпечення моніторингу є конфігурація мікроконтролерів, термопідвісок як датчиків в контексті елеватора, і програмного забезпечення. На противагу ПЛК, використання мікроконтролерів є менш витратним з точки зору ресурсів рішенням, та водночас цілком задовольняє вимогам системи моніторингу.

У цій роботі розглядається апаратно-програмний комплекс моніторингу температури зерносховищ «SmartTerm2000» (надалі скорочено апаратно-програмний комплекс), який вже впроваджено на ряді елеваторів для отримання показників з термопідвісок, які містять цифрові датчики температури. В стандартній комплектації основні компоненти комплексу є наступними.

а) Апаратні:

1) термопідвіска ТП12 (до декількох десятків шт. на круглий силос), в яку входять цифрові датчики температури типу DS18B20;

2) контролер MB-1W24 (в більшості випадків 1 шт. на круглий силос), який опитує із заданим періодом підключені до нього

термопідвіски, зберігаючи значення температури у регістрах. Водночас контролер підключається до комп'ютера оператора через інтерфейс RS485.

б) Програмні, які встановлюються на комп'ютер оператора:

1) програмне забезпечення (ПЗ) «SmartTermRegistrator» (далі Registrator), яке дозволяє із заданим періодом автоматично зчитувати показники температури з регістрів контролера MB-1W24, та зберігати їх у базі даних на комп'ютері оператора;

2) ПЗ SmartTermViewer (далі Viewer), яке використовує візуальні компоненти для відображення максимальної температури датчиків кожної з підвісок із позначенням розташування цієї підвіски в межах зерносховища, для завантаження показників Viewer звертається до системи керування базами даних (СКБД) Firebird.

Метою цієї роботи є підвищення ефективності апаратно-програмного комплексу моніторингу температурного режиму зерносховищ, який вже впроваджено на ряді підприємств, шляхом розробки його додаткових елементів.

Комплекс моніторингу «SmartTerm2000» стикається з двома проблемами, вирішення яких дозволить підвищити його ефективність, а саме:

– відсутність підтримки контролером MB-1W24 роботи із термопідвісками, в які входять аналогові датчики;

– потрібне рішення для полегшення моніторингу окремих силосів (зерноскладів), так як наявний графічний інтерфейс ПЗ Viewer надає недостатньо інформації про схематичне розташування датчиків, що показують температуру поза межами допустимого діапазону. До того ж бажано надати можливість більш зручного перегляду показників кожного датчика у термопідвісці.

Актуальність роботи полягає в економічній вигідності модернізації комплексів термометрії та адаптації застарілих елементів. До того ж розробка нових або покращення наявних елементів надає нові можливості операторам підприємства та підвищує якість збереженого зерна.

Прикладами аналогів комплексу SmartTerm є імпортні системи StorMax OPI-SYSTEM та Rolfes BOONE.

StorMax включає температурні сенсорні кабелі, які програмуються окремо, а також управляючий контролер для збору даних з усіх датчиків. Показники відправляються на комп'ютер оператора, де візуалізуються за допомогою спеціалізованої комерційної програми аналога Viewer.

Комплекс Rolfes BOONE має схожу до StorMax структуру та функціонування, а також дозволяє відправляти сигнали тривоги про вихід температури за допустиму межу засобами SMS або на електронну пошту.

Переваги SmartTerm та доцільність розробки додаткових елементів комплексу можна аргументувати:

– ефективністю та детальністю моніторингу температурного режиму;

– підтримкою різних типів датчиків температури та універсальністю обробки їх показників на верхньому рівні і на етапі візуалізації;

– економічною вигідністю поряд з конкурентами.

1 ПОСТАНОВКА ЗАДАЧІ

Практичною частиною цієї роботи є проектування та реалізація елементів апаратно-програмного комплексу. Створення нових або покращення вже наявних елементів здійснюється для подолання проблем зазначених раніше. У цьому розділі увага зосереджується на описі обраного шляху вирішення проблем та важливих деталях предметної області.

На схемі загальної структури апаратно-програмного комплексу (рис. 1.1) модуль, для якого розробляється прошивка, зображений на червоному фоні, а ПЗ Viewer на синьому.

Відсутність підтримки аналогових датчиків може бути компенсована проектуванням модуля на основі контролера, який візьме на себе функцію програмної емуляції взаємодії контролера верхнього рівня MB-1W24 з термопідвіскою аналогових датчиків за принципом прирівнювання аналогових датчиків до цифрових DS18B20. Водночас передача даних між MB-1W24 та модулем, що проектується, буде здійснюватися за протоколом 1-Wire, де MB-1W24 буде ведучим пристроєм.

Надалі модуль, що проектується, буде називатися 1W-Rx6. Назва складається з двох частин: «1W» означає протокол 1-Wire, а «Rx6» – 6 одиниць аналогових термодатчиків (терморезисторів). Аргументацію актуальності та результати розробки прошивки для модуля 1W-Rx6 також викладено у тезах (додаток В) до XXII Всеукраїнської конференції студентів та молодих науковців [1].

Питання адаптації сучасних апаратно-програмних комплексів до роботи з аналоговими датчиками температури є важливим, оскільки на території країни продовжує функціонувати певна кількість силкорпусів радянської забудови, які як обладнання використовують аналогові термопідвіски вироблені по стандартам минулих років.

Деталізація показників температури у Viewer планується через впровадження у ПЗ можливості створювати додаткові графічні вікна для

кожного з зерносховищ круглої або прямокутної форми, де одним з основних елементів буде таблиця, в якій підвіскам відповідатимуть стовпці, а датчикам – рядки. Використання вже реалізованої функції температурного градієнта дозволить розфарбовувати комірки таблиці залежно від значення температури (див. розділ 4.1). Передбачається, що зміни у ПЗ також дозволять здійснювати користувацькі налаштування для кожного з графічних вікон, що представляють силоси.

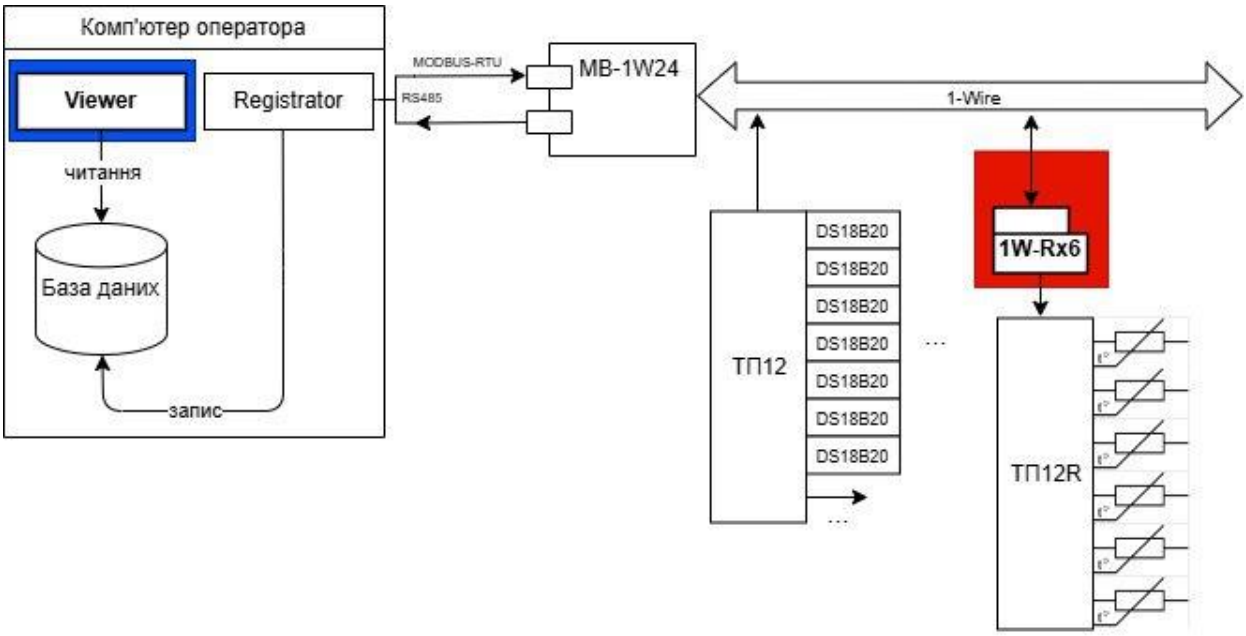

Рисунок 1.1 – Структура елементів апаратно-програмного комплексу

Актуальність роботи полягає в економічній вигідності модернізації комплексів термометрії та адаптації застарілих елементів. До того ж розробка нових або покращення наявних елементів надає нові можливості операторам підприємства та підвищує якість збереженого зерна.

Прикладами аналогів комплексу SmartTerm є імпортні системи StorMax OPI-SYSTEM та Rolfes BOONE, та українська Temix («Темікс»).

Система StorMax включає температурні сенсорні кабелі, які програмується окремо, або аналогові датчики типу термопара. Управляючий контролер зчитує показники з усіх датчиків, а потім ці показники

відправляються на комп'ютер оператора, де візуалізуються за допомогою спеціалізованої комерційної програми аналога Viewer. Комплекс Rolfes BOONE має схожу до StorMax структуру та функціонування, та також дозволяє працювати із датчиками типу термопара. Перевагою ПЗ Viewer будуть можливість задавати користувацькі налаштування на рівні окремого зерносховища та детальність моніторингу.

Частковим аналогом модуля, що розробляється, є контролер компанії Темікс «БІТ-12М», який вимірює показники терморезисторів, перетворює їх у цифрову форму та відправляє на верхній рівень. Однак на відміну від модуля 1W-Rx6 до контролера-аналога підключається велика кількість підвісок, коли до 1W-Rx6 одна. На практиці при підключенні до комплексів термометрії у 1W-Rx6 будуть переваги завдяки таким факторам:

– легкості монтажу обладнання через відсутність потреби у додаткових кабелях – модуль поміститься і у блок комутації підвіски;

– економічній вигідності при модернізації: при заміні аналогових датчиків на цифрові потрібно буде лише підключити цифрові датчики до контролера верхнього рівня. Тим часом як компанія Темікс змушена буде повністю міняти апаратну частину системи.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Протокол 1-Wire

Протокол 1-WIRE передбачає живлення та передачу даних по одній лінії у схемі з одним ведучим та багатьма веденими пристроями. Ведучим пристроєм зазвичай виступає мікроконтролер (МК), який розпочинає сеанс зв'язку, передає дані, а також дозволяє веденим пристроям їх відправку. Прикладом веденого пристрою, що підтримує протокол 1-Wire, є цифровий датчик температури DS18B20. Цей датчик входить у склад термопідвісок та виступає еталоном для емуляції роботи цифрових датчиків за допомогою модуля 1W-Rx6, який проектується.

1-Wire забезпечує гнучкість та ефективність у рішеннях, де задовольняє стандартна швидкість передачі у 16,4 Кбіт/с [2]. Якщо номінал підтягуючого резистора не менше 1 кОм, то на лінію може бути підключено до 30 пристроїв, при її довжині понад 200 метрів.

Протокол 1-Wire підтримують наступні апаратні компоненти комплексу: контролер MB-1W24, цифрові датчики температури DS18B20, модуль 1W-Rx6. У схемах підключення контролер MB-1W24 виступає ведучим пристроєм, датчики DS18B20 та модулі 1W-Rx6 – веденими.

Передача даних між ведучим та усіма сімействами ведених пристроїв за протоколом 1-Wire є однаковою за винятком декількох функціональних команд. Під час передачі сигналів по шині 1-Wire вся інформація сприймається пристроями або як команди, або як дані.

Будь-який обмін даними на шині починається з імпульсу «скидання» або «RESET», який створюється ведучим МК. Цей імпульс є низьким рівнем на лінії протягом мінімум 480 мікросекунд (мкс), але не більше 920 мкс.

Впізнавши імпульс скидання, ведені пристрої очікують від 15 до 60 мкс після його завершення (тобто після появи високого рівня), і притягують лінію

до логічного нуля на 60-240 мкс, що підтверджує їхню присутність на лінії. Такий імпульс називається «імпульсом присутності» або «PRESENCE». Незалежно від того, один чи більше датчиків підключено до лінії, ведучий МК отримує один імпульс присутності. Нижче наведено часову діаграму імпульсів скидання та присутності (рис. 2.1) [2].

Рисунок 2.1 – Часова діаграма сигналів RESET та PRESENCE

Наступним етапом на шині 1-Wire є передача інформаційних бітів від ведучого пристрою до веденого та у зворотному напрямку. «*…обмін інформацією ведеться так званими тайм-слотами: один тайм-слот служить для обміну одним бітом інформації*» [2]. Передача будь-якого тайм-слоту на шині починається з низького рівня, який встановлює ведучий МК.

Якщо від ведучого МК до датчика передається логічний 0, то ведучий МК залишає лінію даних притягнутою до землі на інтервал до 120 мкс. Якщо МК передає логічну 1, то після 1 мкс низького рівня на лінії МК відпускає її. Повинно пройти близько 15 мкс перед тим як датчики впізнають рівень на лінії.

Якщо інформаційні біти відправляє датчик, то обмін все одно починається з інтервалу 1 мс низького рівня на лінії, який створює ведучий МК. При передачі логічного 0, передавши інтервал низького рівня, ведучий

МК відпускає шину, а через 1-2 мкс вже датчик притягує шину до нуля. У передачі логічної 1 після того, як МК відпустив шину, датчик не притягує її до нуля.

Часові діаграми передачі інформаційних бітів по протоколу 1-Wire зображені на рис. 2.2 [2].

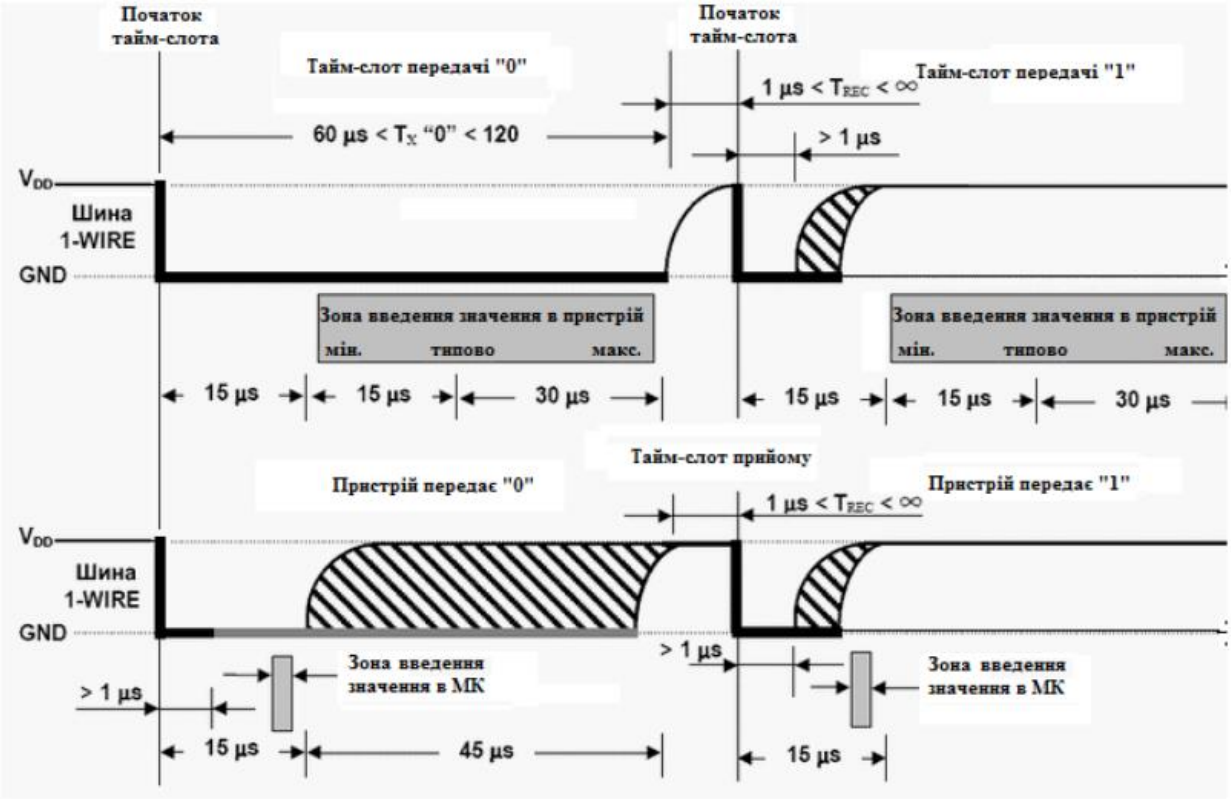

Рисунок 2.2 – Часові діаграми передачі даних в мережі 1-WIRE

2.2 Датчик DS18B20

Інтегральна схема DS18B20 є цифровим датчиком для вимірювання температури у градусах Цельсія з програмованою точністю від 9 до 12 біт. Передача даних від DS18B20 до центрального мікроконтролера здійснюється по шині 1-Wire, яка вимагає лише однієї лінії передачі даних (і землі) для зв'язку. Водночас, DS18B20 живитися безпосередньо від лінії передачі даних («паразитне живлення»), що усуває необхідність зовнішнього джерела

живлення. Загалом вимірювання температури здійснюється у діапазоні від -55 до $+125℃$ при збереженні точності $\pm 0{,}5℃$ у діапазоні від -10 до $+85℃$.

Кожен датчик DS18B20 містить постійний запам'ятовуючий пристрій (ROM), який зберігає його унікальну 8-байтну адресу. Також пам'ять датчика включає 2-байтний регістр температури, 1-байтний регістр конфігурації, CRC-код попередніх 8 байтів у окремому регістрі тощо. Усього пам'ять датчика налічує 9 байтів.

Результат вимірювання температури зберігаються як 16-бітне число з доповненням до двох знаків у 2-байтному регістрі температури. Формат регістру температури наведено на рисунку 2.3 [3].

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

Рисунок 2.3 – Формат регістру температури датчика DS18B20

Програмна реалізація взаємодії ведучого МК з модулем 1W-Rx6 по протоколу 1-Wire наближена до взаємодії ведучого МК з термопідвіскою датчиків DS18B20. На початку виконання прошивки контролера 1W-Rx6 емулюється запис значень за замовчуванням згідно з даташитом в регістри пам'яті цифрових датчиків.

Передача даних від контролера верхнього рівня MB-1W24 до ведених пристроїв передбачає відправку команд та адрес. При емуляції адреси аналогових датчиків представлені в наступному форматі: 28NNXXFFFFFFFFRR, де 28 – номер сімейства; NN – номер підвіски; XX – номер датчика в підвісці; RR – CRC-код адреси.

«*Команди мікроконтролера діляться на дві групи:*

– ROM-команди – команди для роботи з адресами пристроїв (пошук адреси, зчитування адреси, вибір адреси, тощо);

– функціональні команди – команди, наявність яких вимагає виконання відповідних дій з боку кінцевого пристрою» [2].

Нижче у таблиці 2.1 [2] наведено назви, коди та призначення ROM-команд.

Таблиця 2.1 – ROM-команди пристроїв 1-WIRE

Команда	Значення байта	Опис
SEARCH ROM	0xF0	Пошук адрес – використовується при універсальному алгоритмі визначення кількості та адрес підключених пристроїв
READ ROM	0x33	Зчитування адреси пристрою - використовується для визначення адреси єдиного пристрою на шині
MATCH ROM	0x55	Вибір адреси – використовується для звернення до конкретної адреси пристрою з багатьох підключених
SKIP ROM	0xCC	Ігнорування адреси - використовується для звернення до всіх або єдиного пристрою на шині, при цьому адреса пристрою ігнорується (можна звертатися до невідомого пристрою)

Послідовність команд ведучого МК до датчика DS18B20 та до контролера 1W-Rx6 є однаковою за виключенням адрес ведених пристроїв. Послідовність функціональних команд, які обробляє 1W-Rx6 є наступною:

1) команда перетворення (вимірювання) температури (код 0x44);

2) команда читання байтів з оперативної пам’яті датчика (код 0xBE);

3) відправлення байтів з ОЗП датчика до ведучого МК.

Спільне виконання команд 2-3 вимагає скидання 1W-Rx6 командою RESET та повторення усіх дій до відправки адреси включно. Тобто, команди вимірювання та читання температури з датчика мають бути у різних сеансах зв'язку.

2.3 Архітектура мікроконтролера ATtiny44

В основі модуля 1W-Rx6 лежить мікроконтролер ATtiny44, який більшою мірою визначає його можливості. «*Восьмибітні однокристальні мікроконтролери (ОМК) AVR виробництва Atmel призначені, в основному, для вмонтуємих додатків. Вони виготовляються по КМОН-технології, що у сукупності з удосконаленою RISC-архітектурою дозволяє досягти найкращого співвідношення вартість/швидкодія/енергоспоживання*» [4]. Це дозволяє мікроконтролеру наближатися до 1 MIPS на кожен МГц тактової частоти.

Організація пам'яті мікроконтролерів сімейства AVR виконана по Гарвардській архітектурі, в якій розподілені не тільки адресні простори для пам'яті програм та пам'яті даних, але також і шини доступу до них.

Пам'ять програм призначена для зберігання команд, що керують функціонуванням контролера, та констант, які не змінюються під час роботи програми. У контролері ATtiny44 пам'ять програм виконана у вигляді енергонезалежної перепрограмованої Flash-пам'яті обсягом 4 Кбайт. Оскільки усі команди займають в пам'яті 16-біт, то пам'ять програм має 16-розрядну організацію.

Пам'ять даних мікроконтролера ATtiny44 поділена на три частини:

– регістрову пам'ять;

– ОЗП (англ. SRAM – статичний ОЗП);

– енергонезалежну пам'ять, що очищується та програмується за допомогою електрики (EEPROM).

Регістрова пам'ять знаходиться в адресному просторі ОЗП, що складається з 8-бітних сторінок. З огляду на це перші 96 байтів

(у шістнадцятковій системі числення 0x00–0x5F) представляють адреси регістрів. Залишок адресного простору становить 256 байт та найчастіше і розуміється під ОЗП.

Регістрова пам’ять функціонально поділяється на:

– 32 регістри загального призначення (РЗП), які можуть зберігати як дані, так і адреси (0x00–0x1F);

– 64 регістри вводу-виводу або регістри внутрішніх пристроїв (0x20 – 0x5F). Мають визначений виробником двійковий формат, та використовуються для налаштування режимів роботи, внутрішніх пристроїв, дозволі переривань та відображені станів мікроконтролера.

Пам’ять даних EEPROM обсягом 256 байт знаходиться у окремому адресному просторі. Цей тип пам’яті доступний для читання і констант, які можуть представляти ідентифікаційні дані, налаштування та інше.

Важливим компонентом мікроконтролера також є АЛП (арифметико-логічний пристрій), який виконує арифметичні, логічні та бітові операції. Після виконання операції регістр стану SREG оновлюється для відображення ознак результату операції. Надалі біти регістру стану можуть використовуватися для зміни ходу виконання програми за допомогою команд умовних переходів, що підтримуються.

В роботі застосовуватимуться біти I, Z, C що позначають глобальний дозвіл переривань, нульовий результат арифметичної операції та перенесення відповідно.

Порти A та B мікроконтролера ATtiny44 включають 12 ліній вводу-виводу загального призначення, які програмно конфігуруються або на вхід, або на вихід. Для цього застосовується регістр DDRA (напрямку даних) та аналогічний для порту B (DDRB). До усіх ліній можуть бути підключені вбудовані резистори, що підтягують. Кожна лінія крім загальних функцій виконує спеціальні для неї, які залежать від підключених до неї внутрішніх пристроїв. Схему ліній вводу-виводу ATtiny44 наведено на рис. 2.4 [5].

Наступні компоненти мікроконтролера використовуватимуться у роботі, та налаштовуватимуться за допомогою регістрів внутрішніх пристроїв:

– 8-бітний таймер/лічильник;

– внутрішні та зовнішні джерела переривань;

– вбудований 8-канальний 10-розрядний аналого-цифровий перетворювач (АЦП);

– програмований сторожовий таймер з внутрішнім генератором тактової частоти;

– основний внутрішній генератор;

– три режими зниженого енергоспоживання з програмним переходом;

– аналоговий компаратор.

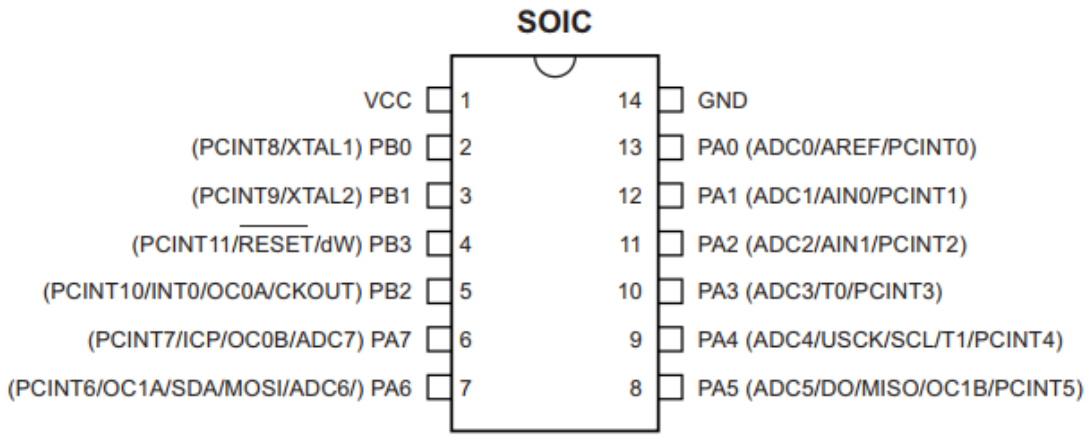

Рисунок 2.4 – Схема ліній вводу-виводу Attiny44 у корпусі SOIC

3 СТЕК ТЕХНОЛОГІЙ, ЩО ВИКОРИСТОВУЮТЬСЯ

3.1 Інструменти відладки та програмування прошивки Attiny44

Для редагування та відладки коду прошивки контролера Attiny44 використовувалося IDE Atmel Studio 7.0 (до 6 версії відоме як AVR Studio). Обрано саме це середовище розробки, оскільки ПЗ підтримує емуляцію роботи цілого ряду мікроконтролерів (серед яких є і Attiny44), дозволяє під час відладки спостерігати за станом прапорів, лічильника тактів та різних типів пам’яті, змінювати логічні рівні на входах мікроконтролера. Як і більшість IDE, Atmel Studio підтримує механізм точок зупинки (breakpoints).

Результатом компіляції коду проекту у Atmel Studio є бінарний файл, який завантажуватиметься (прошиватиметься) у Flash-пам’ять мікроконтролера. Для завантаження бінарного файлу прошивки у мікроконтролера використовується спеціалізований пристрій – програматор. При підключенні програматора до USB-порту комп’ютера це з’єднання емулюється як стандартний COM-порт.

На платі програматора розташовується роз’єм для внутрішньосхемного програмування, що включає лінії, які задіяні у протоколі SPI (MISO, MOSI, RST, SCLK). Саме цей протокол використовується для запису бінарного файлу прошивки у Flash-пам’ять під час програмування. Певні входи мікроконтролера Attiny44 виконують функції ліній протоколу SPI: PA5 – MISO, PA6 – MOSI, PB3 – RESET, PA4 – SCLK.

Для прошивки мікроконтролера використовується мультисистемний програматор-адаптер на мікросхемі FT232HQ (рис. 3.1). Для зручності підключення контролера до програматора в процесі прошивання використовується панель з нульовим зусиллям, куди вставляється сам контролер.

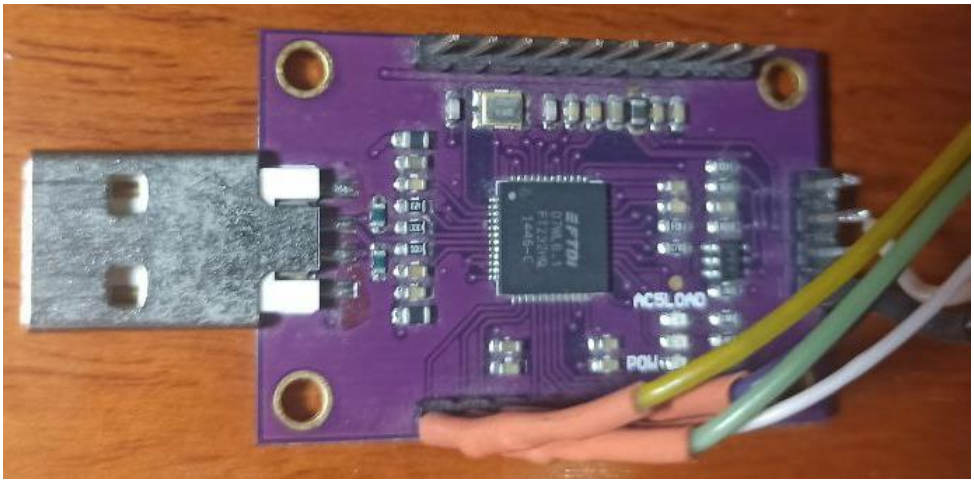

Рисунок 3.1 – Плата програматора на базі FT232HQ, що використовується

3.2 Технологія роботи з СКБД Firebird

ПЗ Viewer під час роботи звертається до бази даних (БД) Firebird для доступу до значень температури датчиків на певну дату та час. Firebird є вільною реляційною клієнт-серверною СКБД, яка пропонує значну кількість стандартних функцій ANSI SQL. ПЗ Viewer підключається до Firebird, що застосовується по схемі «вбудованого сервера» на Windows.

Вбудований сервер (введений у Firebird 1.5) був спеціально розроблений для полегшення розгортання програм, які постачаються разом із БД Firebird. Встановлення полягає лише в розпакуванні DLL та кількох інших файлів за потрібним розташуванням.

Використання Firebird у режимі вбудованого сервера надає користувацькому застосунку можливість підключатися як до БД, яка розміщена на цьому комп’ютері (надалі вбудована БД), так і до віддаленого сервера БД, використовуючи лише одну бібліотеку fbembed.dll. Згідно з документацією Firebird [7] цей файл потрібно завантажити та перейменувати у fbclient.dll. Унаслідок виходить і клієнтська бібліотека, і вбудований сервер для доступу до вбудованих БД.

3.3 Середовище розробки C++Builder

IDE C++Builder обрано як засіб розробки розширених елементів візуалізації ПЗ Viewer з таких причин:

– для уніфікації підходів та простоти використання наявного коду, оскільки головне вікно програми розроблене також у C++Builder;

– C++Builder підтримує компоненти VCL (Visual Component Library), які дозволяють поєднати мову C++ із простою та швидкою розробкою GUI для прикладного ПЗ;

– підтримці інтерфейсу для роботи із вбудованим сервером Firebird;

– для забезпечення сумісності ПЗ Viewer із попередніми версіями ОС Windows (XP, 7, 8), які з великою ймовірністю можуть бути встановлені на комп’ютері оператора, завдяки простим графічним компонентам на відміну від рішення Visual Studio + WPF.

4 РОЗРОБКА ЕЛЕМЕНТІВ АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ МОНІТОРИНГУ ЗЕРНОСХОВИЩ

4.1 Програмне та апаратне проектування та розробка прошивки для модуля 1W-Rx6

На рис. 4.1 зображено функціональну схему модуля 1W-Rx6, яку отримано на етапі постановки задачі по розробці прошивки для цього модуля.

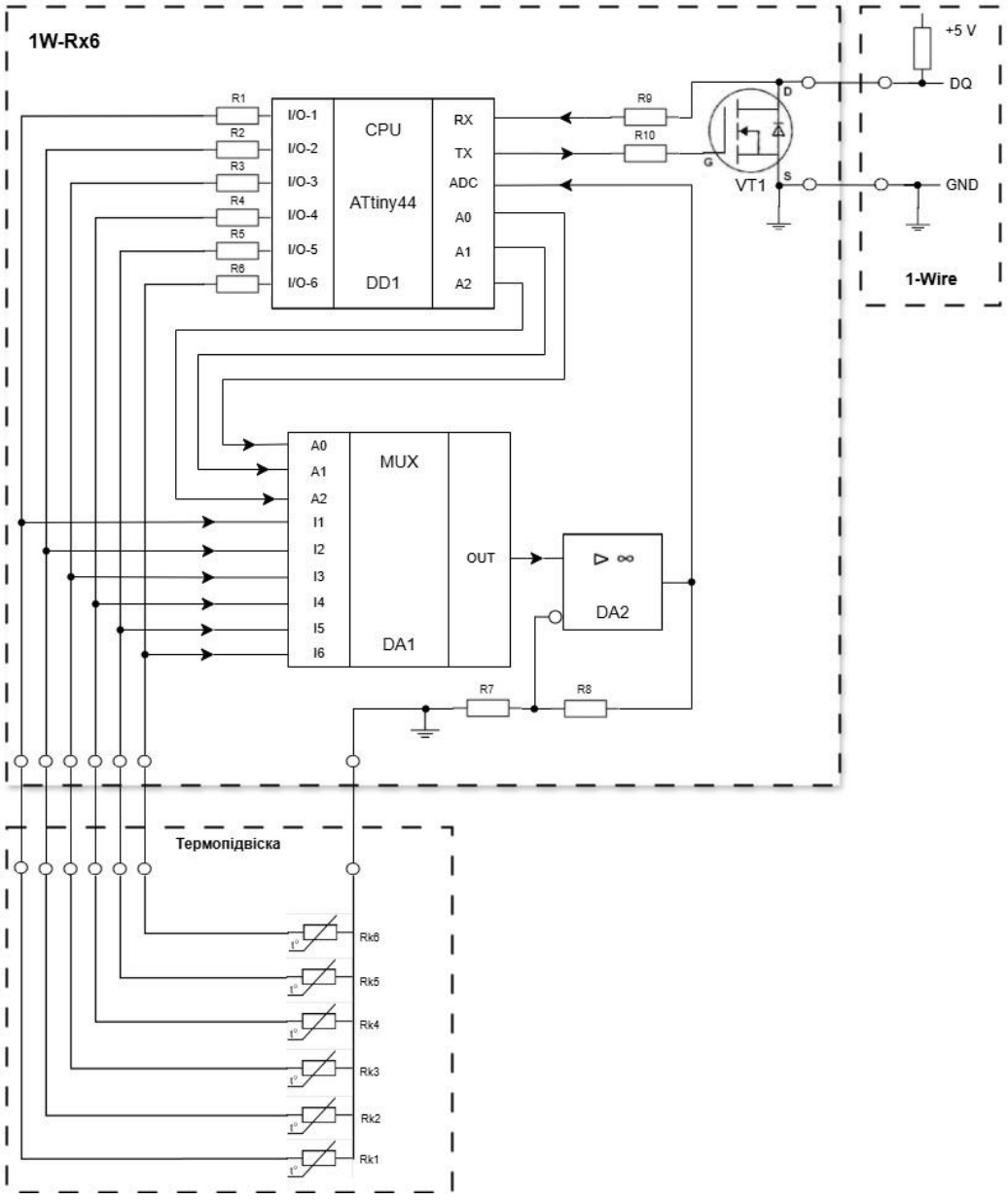

Рисунок 4.1 – Функціональна схема 1W-Rx6

Згідно зі стандартами побудови схем показано напрямки сигналів, роз’ємний тип з’єднання, а також позиційне позначення елементів.

Для налагодження програми використовувався макет плати 1W-Rx6 (рис. 4.2) із 6-тьма підключеними звичайними резисторами замість мідних терморезисторів, а також резистором для компенсації опору загального проводу. Входами DQ та GND модуль підключається до шини 1-Wire.

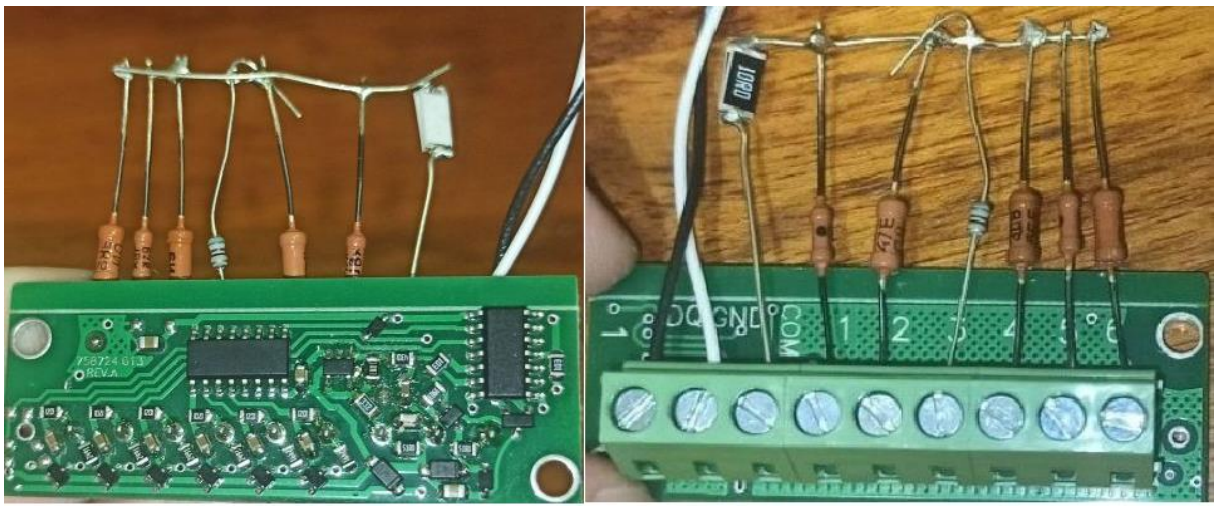

Рисунок 4.2 – Макетна плата та виходи модуля 1W-Rx6

4.1.1 Структура коду прошивки 1W-Rx6

Основним файлом прошивки модуля є main.asm. З міркувань полегшення орієнтації у програмі її код поділений на фрагменти віднесені в окремі файли, такі як: init, res_pres, recv_addr, measure_temp, send_temp, які виконуються послідовно та відповідають початковим налаштуванням пристрою, обробці подій на шині 1-Wire і функціональним командам датчика DS18B20. Також відокремленими є файли обробників апаратних переривань, перейменування регістрів та таблиці (масиви) констант. Усі фрагменти включаються у файл main.asm директивою #include. До того ж у цьому файлі є вектори переривань та мітка, яка вказує на адресу першої інструкції (0x00), що еквівалентно початку програми.

Серед включень:

– файл init.inc, код якого виконується при подачі живлення;

– фрагменти тіла програми: init.inc, res_pres.inc, recv_addr.inc, measure_temp.inc, send_temp.inc;

– обробники переривань: RES_PRES_1W.inc, RES_PRES_TIME.inc, BIT_1W.inc, BIT_TIME.inc;

– перейменування в рамках проекту – def.inc;

– перейменування регістрів і бітів портів вводу-виводу та внутрішніх пристроїв МК сімейства AVR – tn44adef.inc;

– макроси, що використовуються в проекті – macro.inc;

– перейменування та змінні в ОЗП – var.inc;

– константи та таблиці (масиви) у ПЗП - tab.inc.

Вихідний код перелічених модулів зберігається на диску, що додається до роботи.

4.1.2 Основне тіло програми

Основне тіло в контексті програми («прошивки») модуля 1W-Rx6 передбачає програмний код без обробників переривань. В рамках роботи обробляються апаратні переривання, які викликаються або подіями у внутрішніх пристроях мікроконтролера, або подіями на шині 1-Wire.

Тілу програми відповідає ліва частина основної блок-схеми (рис. 4.3), тобто фрагменти init.inc, res_pres.inc, recv_addr.inc, measure_temp.inc, send_temp.inc. В рамках тіла програми:

– емулюється сеанс роботи ведучого з цифровими датчиками DS18B20 по протоколу 1-Wire при підключенні до модуля 1W-Rx6 аналогових терморезисторів у дійсності: отримання сигналу «Скидання», передача сигналу «Присутність»;

– виконуються функціональні команди датчика DS18B20: вимірювання температури, відправка показників ведучому на шині 1-Wire.

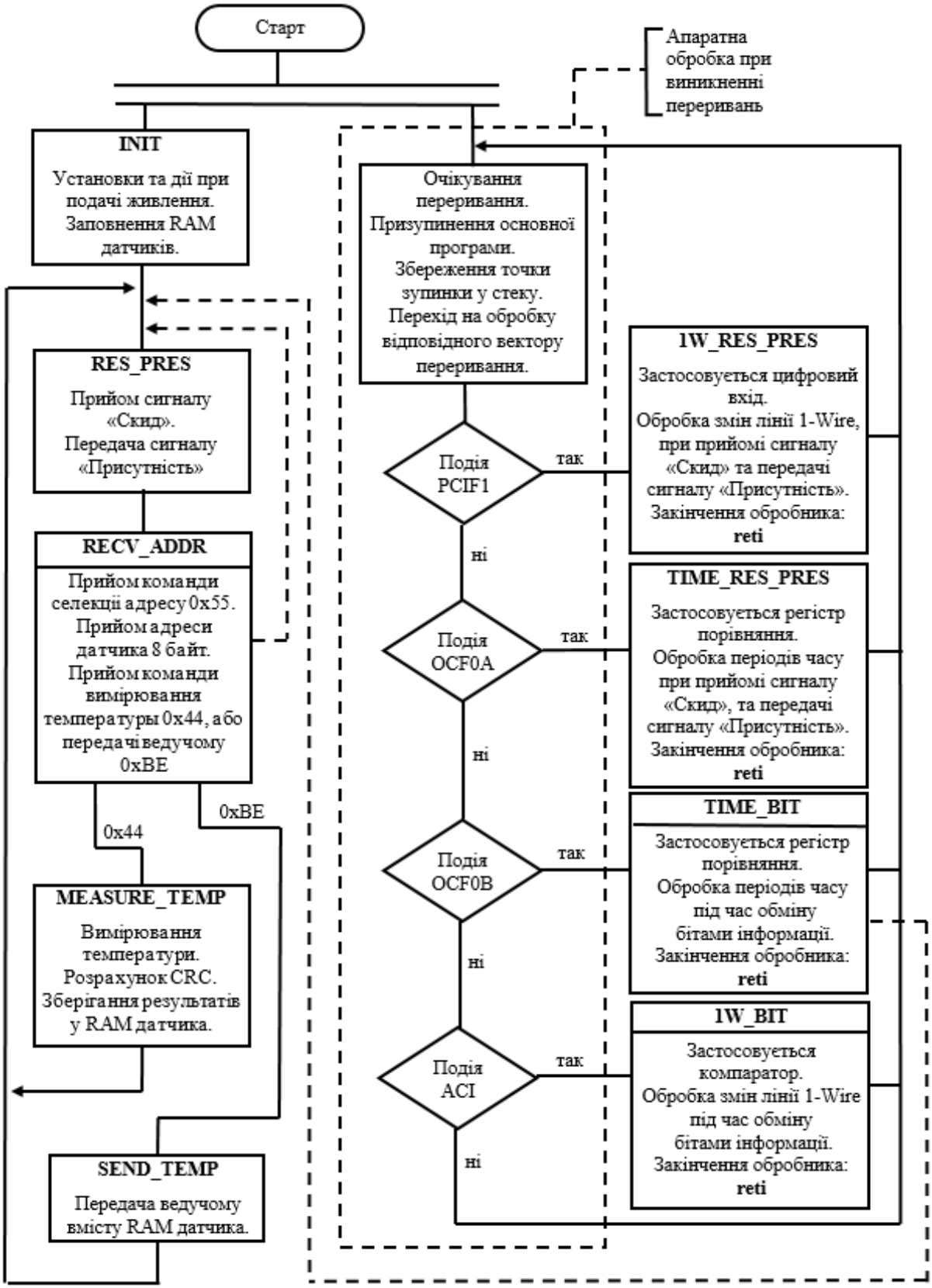

Рисунок 4.3 – Основна блок-схема програми

Частина основної блок-схеми, що виділена пунктирною лінією включає апаратні дії контролера та джерела переривань як умови переходу на обробники.

Зміна ознак сеансу, прийом та відправка інформаційних бітів по шині 1-Wire виконується поза кодом основного тіла в обробниках переривань, які відображені у правій частині основної блок-схемі. В тих фрагментах основного тіла програми, що відповідають етапам прийому та відправки бітів, розміщено цикли, з виходом за умовою установки певної ознаки. Наприклад, при отриманні коду функціональної команди від ведучого на шині 1-Wire очікуються усі 8 біт цього коду, кожен з яких надсилається по порядку, тобто лічильник приймача RxCnt має дорівнювати 8.

Під час створення блок-схем основним нормативним документом був ДСТУ ISO 5807:2016, який спирається на ISO 5807:1985.

4.1.3 Опис окремих блоків програми

Програма починається з виконання включеного файлу init.inc. У цьому файлі виконуються початкові установки внутрішніх пристроїв, налаштування портів вводу-виводу, завантаження значень за замовчуванням у сторінки усіх 13-ти датчиків в ОЗП модуля, розрахунок CRC-кодів цих значень. Фізично до модуля підключаються 6 датчиків, а контролер ATtiny44 у складі модуля 1W-Rx6 включає ще 1 внутрішній. Концептуально температурне перетворення з аналогового датчика відбувається або з компенсацією опору проводів, або без. Тому програмно кожному аналоговому датчику (крім вбудованого) відповідає два окремих. Серед важливих установок у файлі init.inc:

а) зниження енергоспоживання модуля під час очікування сигналу «Скидання» шляхом збільшення дільника тактової частоти МК (дільник 8) та дозволу входу в енергозберігаючий «idle» режим;

б) конфігурування ліній, до яких підключені датчики, адресні входи мультиплексора [6] та лінії передачі Tx на вихід;

в) конфігурування лінії приймання Rx на вхід;

г) в якості опорної напруги АЦП визначається напруга джерела живлення;

д) на позитивний (прямий) вхід аналогового компаратора підключається опорна напруга, а на інверсний – приймач 1-Wire.

У файлі res_pres.inc реалізовано прийом сигналу «Скидання», передачу сигналу «Присутність» та установки для очікування подій переривань.

Блок recv_addr виконується після коду у файлі res_pres.inc та реалізує прийом наступних байтів:

– команди розпізнавання адреси (код 0x55);

– адреси датчика типу DS18B20 до якого звертається ведучій;

– коду функціональної команди (0x44 або 0xBE).

В рамках блоку recv_addr також виконується перехід до виконання наступного блоку в залежності від функціональної команди.

Адреси усіх 13-ти датчиків включно із CRC-кодом, що підключені до модуля 1W-Rx6, зберігаються у постійній пам’яті програм вигляді таблиці, яку зображено на рисунку 4.4.

```
SENSOR_ADDRESS_START:
 //  №00, №01, №02, №03, №04, №05, №06, №07, №08, №09, №10, №11, №12, №00,
.DB 0x55,0x55,0x55,0x55,0x55,0x55,0x55,0x55,0x55,0x55,0x55,0x55,0x55,0x28
.DB      0x28,0x28,0x28,0x28,0x28,0x28,0x28,0x28,0x28,0x28,0x28,0x28
.DB 0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x00
.DB      0x01,0x02,0x03,0x04,0x05,0x06,0xFF,0xFE,0xFD,0xFC,0xFB,0xFA
.DB 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
.DB      0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
.DB 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
.DB      0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
.DB 0xA4,0x69,0x27,0xEA,0xBB,0x76,0x38,0x6D,0xA0,0xEE,0x23,0x72,0xBF,0xFF
```

Рисунок 4.4 – Таблиця адрес датчиків

У коді файлу recv_addr.inc адреса датчика, яка отримана по шині 1-Wire від ведучого, порівнюється з адресами усіх датчиків в таблиці. Якщо знайдено відповідність, то зберігаються номер цього датчика в таблиці та вказівник на його показники в оперативній пам’яті. Надалі в залежності від наступної функціональної команди виконуватиметься або запис за адресою вказівника, або читання. Постійною складовою кожного сеансу передачі даних на шині 1-Wire є передача команди розпізнавання адреса (0x55), тому вона також внесена у таблицю для зменшення розгалужень у коді.

На рис. 4.5 наведено блок-схему алгоритму, що прописаний у файлі recv_addr.inc. Визначення номера датчика у таблиці за адресою, яка прийшла від ведучого, виконується за допомогою двобайтовой маски Addr_mask. Однак використовуються лише 13 біт цієї маски, що відповідають 13-ти датчикам.

У зовнішньому циклі RECV_BYTE приймається черговий байт від ведучого. Цей байт є частиною адреси датчика, якому призначатиметься наступна функціональна команда.

У внутрішньому циклі COMP_TO_SENS_ADR, що виконується 13 разів, байт отриманий від ведучого порівнюється з відповідним байтом адрес усіх датчиків. Якщо для певного датчика не відбулося збігу, то біт маски Addr_mask, що відповідає цьому датчику скидається. Наприкінці з 13-біт маски має залишитися одна одиниця. Порядковим номером цієї одиниці і є номер датчика, до якого звернулися.

У зовнішньому циклі є умова, що перевіряє чи не обнулена уся маска. Якщо маска обнулена, то програма повертається до очікування сигналу «Скидання», так як відбулася помилка або ведучій звертався не до цих датчиків. Надалі разом із номером (індексом) датчика встановлюється вказівник на його дані в оперативній пам’яті.

Наприкінці фрагменту recv_addr приймається функціональна команда та відбувається перехід до її виконання. Якщо код цієї команди не дорівнює 0x44 або 0xBE, то сталася помилка та виконується перехід на очікування сигналу «Скидання».

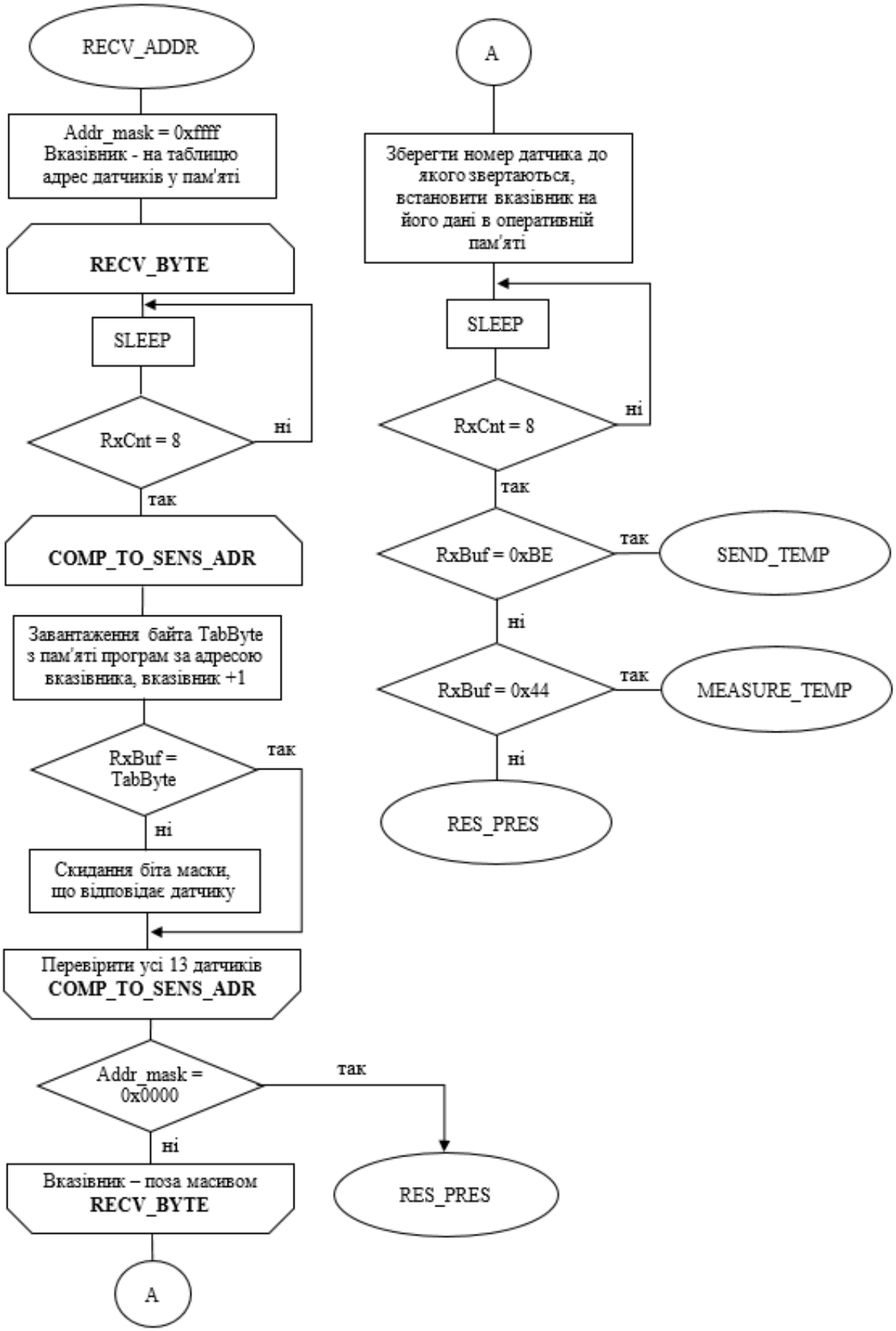

Рисунок 4.5 – Блок-схема алгоритму фрагменту recv_addr

У лістингу 4.1 наведено код блоку recv_addr з коментарями.

```
RECV_ADDR: //Мітка початку блоку
//Формується 2-байтова маска з одиниць
ldi rH_Prg0,0xFF
mov rL_Prg0,rH_Prg0
//Вказівник - на початок таблиці адрес в ПЗП
ldi ZL,Low(2*SENSOR_ADDRESS_START)
ldi ZH,High(2*SENSOR_ADDRESS_START)
RECV_BYTE: //Мітка зовнішнього циклу
//Прийом 8 біт від ведучого
    ldi rH_Prg1,8
      sleep
      nop
      nop
      cp RxCnt,rH_Prg1
  brbs SREG_C,PC-4
  sub RxCnt,rH_Prg1 //Зменшити лічильник приймача на 8
  ldi rH_Prg2,SENSOR_N //Зберегти кількість датчиків у rH_Prg2
  COMP_TO_SENS_BYTE: //Мітка внутрішнього циклу
    lpm rH_Prg1,Z+ //Завантаження байту з ПЗП
  bst rL_Prg0,0 //У прапор T SREG копіюється 0-й біт маски
  cpse rH_Prg1,RxBuf1 /*Порівняння байт з ПЗП та від ведучого
  Якщо рівні, то пропускається наступна команда*/
  // Циклічний зсув 13 бітів маски
  bclr SREG_T
  lsr rH_Prg0
  ror rL_Prg0
  bld rH_Prg0,4 //Вивантажити прапор T у кінець маски
  dec rH_Prg2 //Зменшення лічильника циклу
  brbc SREG_Z, COMP_TO_SENS_BYTE/*Перехід до наступної ітерації,
  якщо лічильник не нуль*/
  /* Перевірка, чи не обнулена маска,
  якщо так - перейти на очікування скидання*/
  and rL_Prg0,rL_Prg0
  brbc SREG_Z,PC+4
  and rH_Prg0,rH_Prg0
  brbc SREG_Z,PC+2
  rjmp RES_PRES
//Якщо досягнутий кінець таблиці - вийти з циклу
```

Лістинг 4.1 – Отримання команди 0x55, отримання адреси датчика, визначення індексу датчика, перехід до виконання функціональної команди

```
cpi ZL,Low(2*SENSOR_ADDRESS_START+SENSOR_N*8)
ldi rH_Prg2,High(2*SENSOR_ADDRESS_START+SENSOR_N*8)
cpc ZH, rH_Prg2
  brbc SREG_Z,RECV_BYTE
PRG_ADR_OK: //Датчик знайдено
//Формується номер датчика та вказівник на його дані у ОЗП
ldi rH_Prg2,0xFF
ldi XL,Low(PAGE_00-SENS_PAGE_SIZE)
ldi XH,High(PAGE_00-SENS_PAGE_SIZE)
adiw XH:XL,SENS_PAGE_SIZE
inc rH_Prg2
lsr rH_Prg0
ror rL_Prg0
brbc SREG_C,PC-4
//Номер датчика та адреса на дані в ОЗП зберігаються
STORE SENSOR_INDEX,rH_Prg2
STORE SENSOR_ADR_L,XL
STORE SENSOR_ADR_H,XH
//Прийом функціональної команди
ldi rH_Prg0,(8)
  sleep
  nop
  nop
  cp RxCnt,rH_Prg0
brbs SREG_C,PC-4
sub RxCnt,rH_Prg0 //Зменшити лічильник приймача на 8
//Перехід в залежності від отриманої команди
mov rH_Prg0,RxBuf1
cpi rH_Prg0,0xBE
   brbs SREG_Z,SEND_TEMP//READ_RAM PRG_ROM_command
cpi rH_Prg0,0x44
   brbs SREG_Z,MEASURE_TEMP//SENSOR convert  PRG_ROM_command
   rjmp RES_PRES //Помилка, повернення до очікування скидання
```

Лістинг 4.1, аркуш 2

4.2 Розробка розширених елементів візуалізації для ПЗ Viewer

4.2.1 Наявна архітектура та можливості ПЗ Viewer

Для деталізації показників температури та можливості змінювати користувацькі налаштування моніторингу створені дочірні вікна для кожного зерносховища, які виділені пунктирною лінією на схемі ієрархії (рис. 4.6).

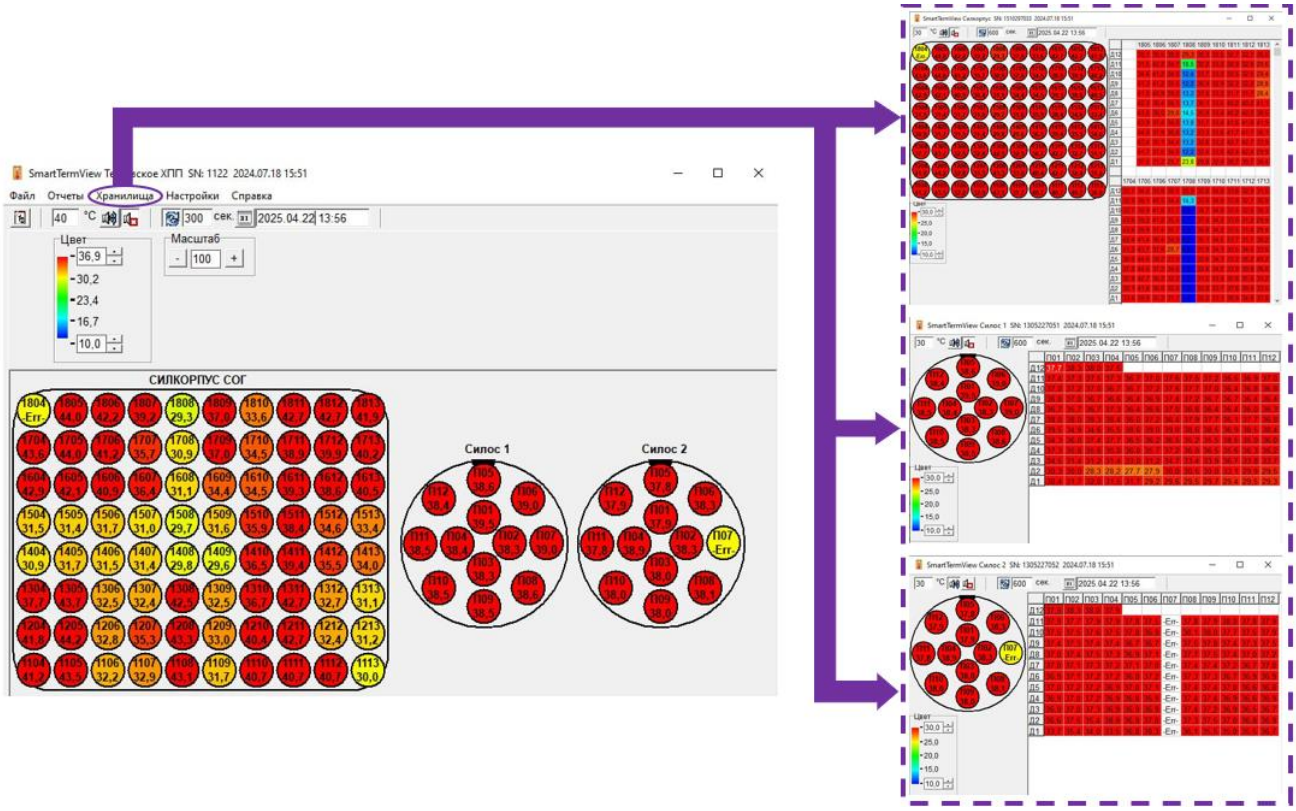

Рисунок 4.6 – Ієрархія віконного інтерфейсу Viewer

Viewer наразі вже виконує функції візуалізації температурного стану об’єкта, сигналізації про вихід температури за допустиму межу, перегляду стану у попередні періоди. ПЗ також дозволяє змінювати ряд користувацьких налаштувань в рамках елеватору (декілька силосів). Розташування елементів управління віконного інтерфейсу, що відповідають за ці налаштування позначено на рис. 4.7.

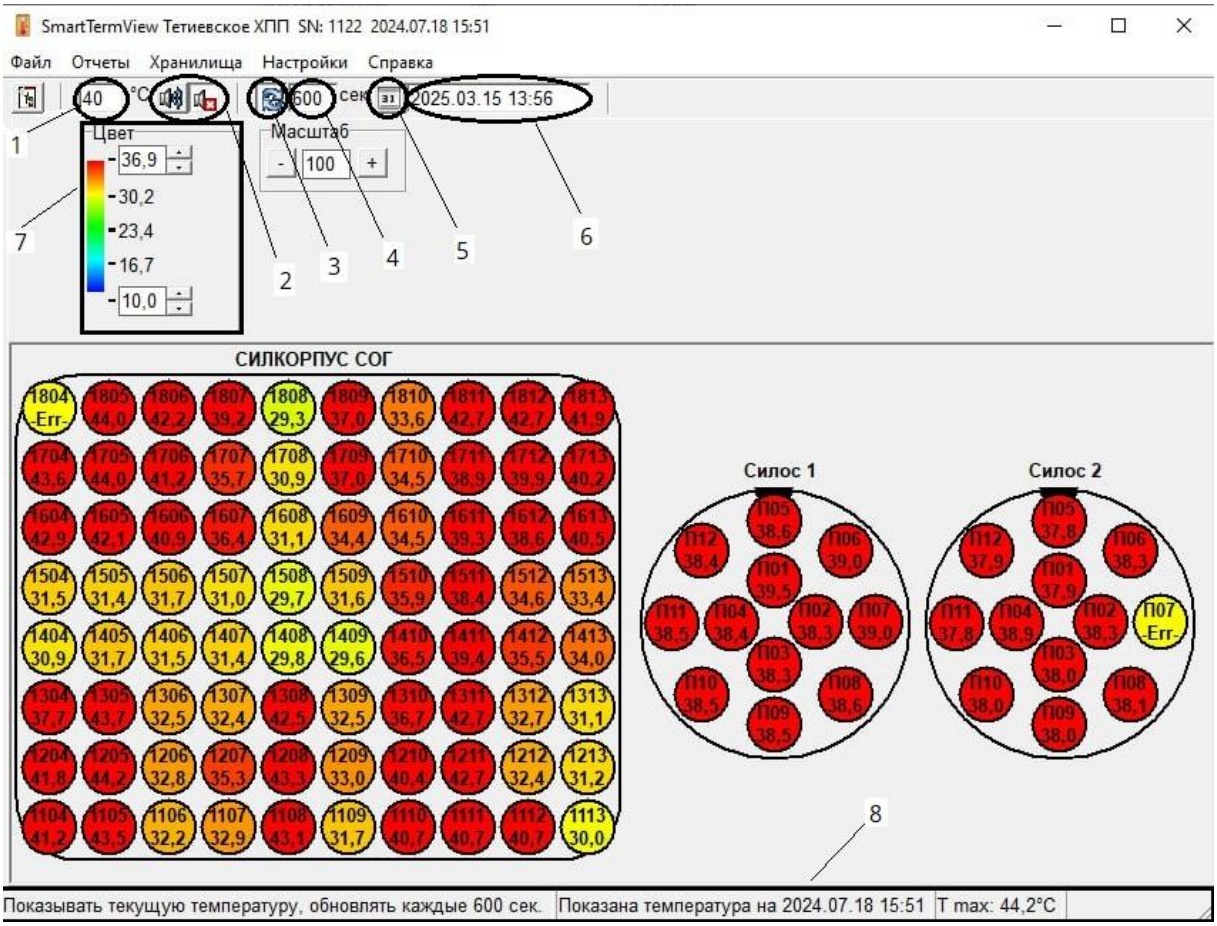

Рисунок 4.7 – Наявний графічний інтерфейс програми Viewer

Примітка:

1 – редаговане текстове поле температури, якщо температура більше, то спрацьовує звуковий сигнал;

2 – пара взаємовиключних кнопок дозволу/заборони звукового сигналу;

3 – кнопка оновлення температури, при натисканні оновлення відбувається в той же момент;

4 – редаговане текстове поле періоду автоматичного оновлення температурного стану;

5 – кнопка ручного оновлення температури на вказану дату та час, є взаємовиключною з кнопкою оновлення температури (пт. 3);

6 – редаговане текстове поле дати та часу на яке відображається температурний стан при ручному режимі;

7 – елемент управління температурним градієнтом, налаштування нижньої та верхньої меж призводить до зміни кольору візуалізації (холодні кольори ставляться у відповідність низькій температурі, теплі - високій);

8 – рядок стану, де відображається максимальна температура та інформація в залежності від режиму.

При запуску Viewer значення користувацьких параметрів зчитуються з файлу STViewer2000v205_US.XML (додаток А). Зміни у параметрах одразу впливають на роботу програми, а при виході з неї викликається допоміжне вікно, яке пропонує зберегти зміни. Залежність програми від файлу користувацьких налаштувань відображена на рис. 4.8. Нижче у лістингу 4.2 наведено тег Object, що відповідає за налаштування батьківського вікна:

```
<Object SerialNumber="1122" TemperatureRed="36,9"
TemperatureBlue="10" TemperatureAlarm="40" AlarmEnable="False"
AutoUpdateDelay="600" ManualDateTime="2024.05.15 13:56"
FormTop="176" FormLeft="842" FormHeight="762" FormWidth="939"
TreeShow="False" WarningDifferenceTime="72" ImageScale="100">
</Object>
```

Лістинг 4.2 – Користувацькі налаштування рівня підприємства з XML-файлу користувацьких налаштувань

Конфігураційний файл STViewer2000v205.XML містить важливу для візуалізації інформацію, а саме:

– структуру елеватора у вигляді дерева (серійний номер елеватора, серійні номери його силосів, серійні номери підвісок у кожному силосі, серійні номери датчиків кожної підвіски);

– типи силосів (круглі, прямокутні), кількість підвісок у силосах, положення, розміри та кут повороту представлень силосів, допоміжну текстову інформацію.

```
<Object Caption="Тетиевское ХПП" SerialNumber="1122"
UseAsDefault="1" UserSettingFileName1="STViewer2000v205_US.XML">
    <Silos SerialNumber="1305227051" Top="100" Left="440"/>
</Object>
<Silos Caption="Силос 1" Address="170" Model="MB1W25"
SerialNumber="1305227051" ROM1="282B3A7202000089">
    <Type TopViewType="Round" BobCount1="4" BobCount2="8"
RotateAngle="0"/>
    <Bob SerialNumber="1305220101" Order="1" TopViewName="01"
Caption="П01"/>
...
    <Bob SerialNumber="1305220112" Order="12" TopViewName="12"
Caption="П12"/>
</Silos>
<Bob SerialNumber="1305220101">
    <Sensor Caption="Датчик01" ROM="280101FFFFFFFF69"
DataBaseName="SENSOR0001"/>
    <Sensor Caption="Датчик12" ROM="280101FFFFFFFF69"
DataBaseName="SENSOR0012"/>
</Bob>
```

Лістинг 4.3 – Приклад структури зерносховища (силосу) з конфігураційного XML-файлу

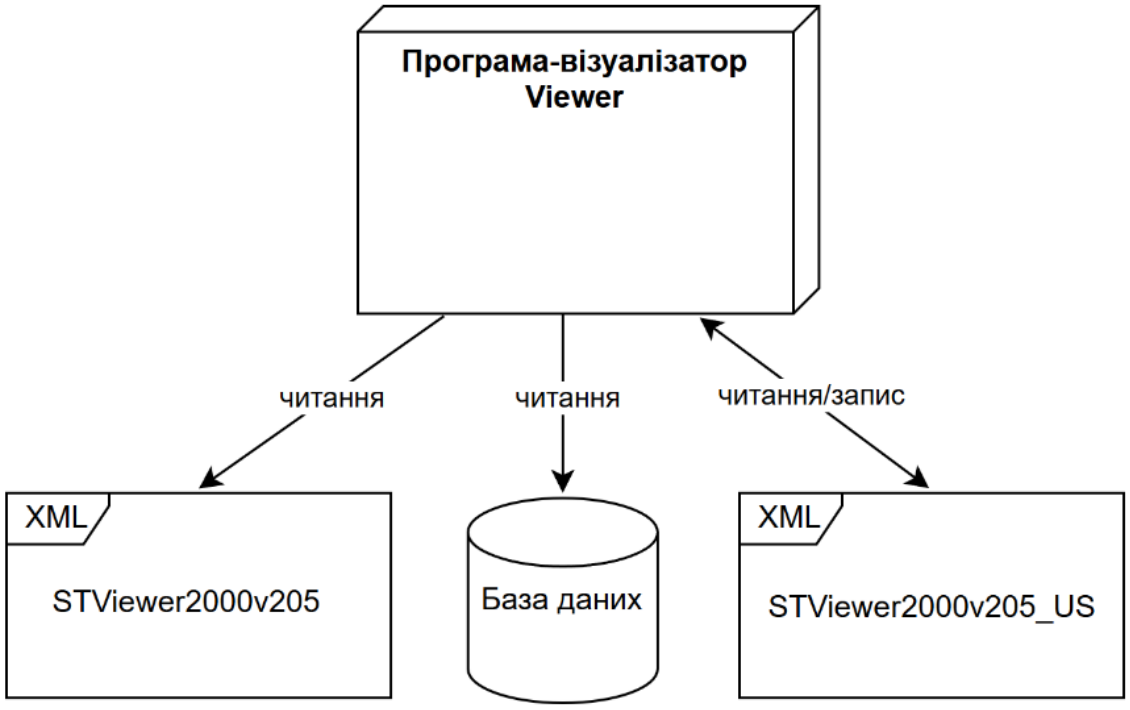

Рисунок 4.8 – Схема залежностей програми Viewer

4.2.2 Класи та методи, що розробляються та використовуються

Оскільки розробка розширення візуалізації Viewer планується шляхом створення вікна, яке матиме інтерфейс аналогічний головному вікну, то частина вже наявних в проекті класів використовуватиметься у цій задачі. Проект можна поділити на класи логіки та класи графічного інтерфейсу. На діаграмі класів ПЗ View (рис. 4.9) блакитним кольором позначено класи першого типу, а рожевим – другого типу. Класи, що розробляються або використовуються у роботі, виділені пунктирною лінією.

Як згадувалося раніше, предметну область програми можна представити як деревовидну структуру: Елеватор (Підприємство) -> Зерносховища (Силоси) -> Підвіски -> Датчики. З огляду на це проект включає відповідні класи логіки Elevator, Silos, Bob та Sensor відповідно. Відношення класів, що представляють батьківські та дочірні вузли дерева, на діаграмі позначені як один до багатьох.

Класи графічного інтерфейсу виконують візуалізацію інформації певних класів логіки. Усі графічні вікна наслідують клас TForm, що є вбудованим класом C++Builder. Головне вікно TFromMain представляє інформацію на рівні всього елеватору, а вікно TFormSilos, що розробляється, представлятиме на рівні певного із зерносховищ. Зміст класу TFormSilos складатимуть методи-обробники подій вбудованих елементів віконного інтерфейсу C++Builder.

Спільними для усіх класів логіки є властивості Caption, SerialNumber та Parent. Ці властивості представляють назву, серійний номер (ідентифікатор) та батьківський вузол відповідно. Для класу Sensor важливим є властивість DBName, яка зчитується з конфігураційного XML-файлу, та являє ім’я датчика у базі даних.

Клас Bob, що представляє підвіски, включає список датчиків SensorList. Властивість TopViewName визначає положення підвіски у силосі зокрема для прямокутних зерноскладів.

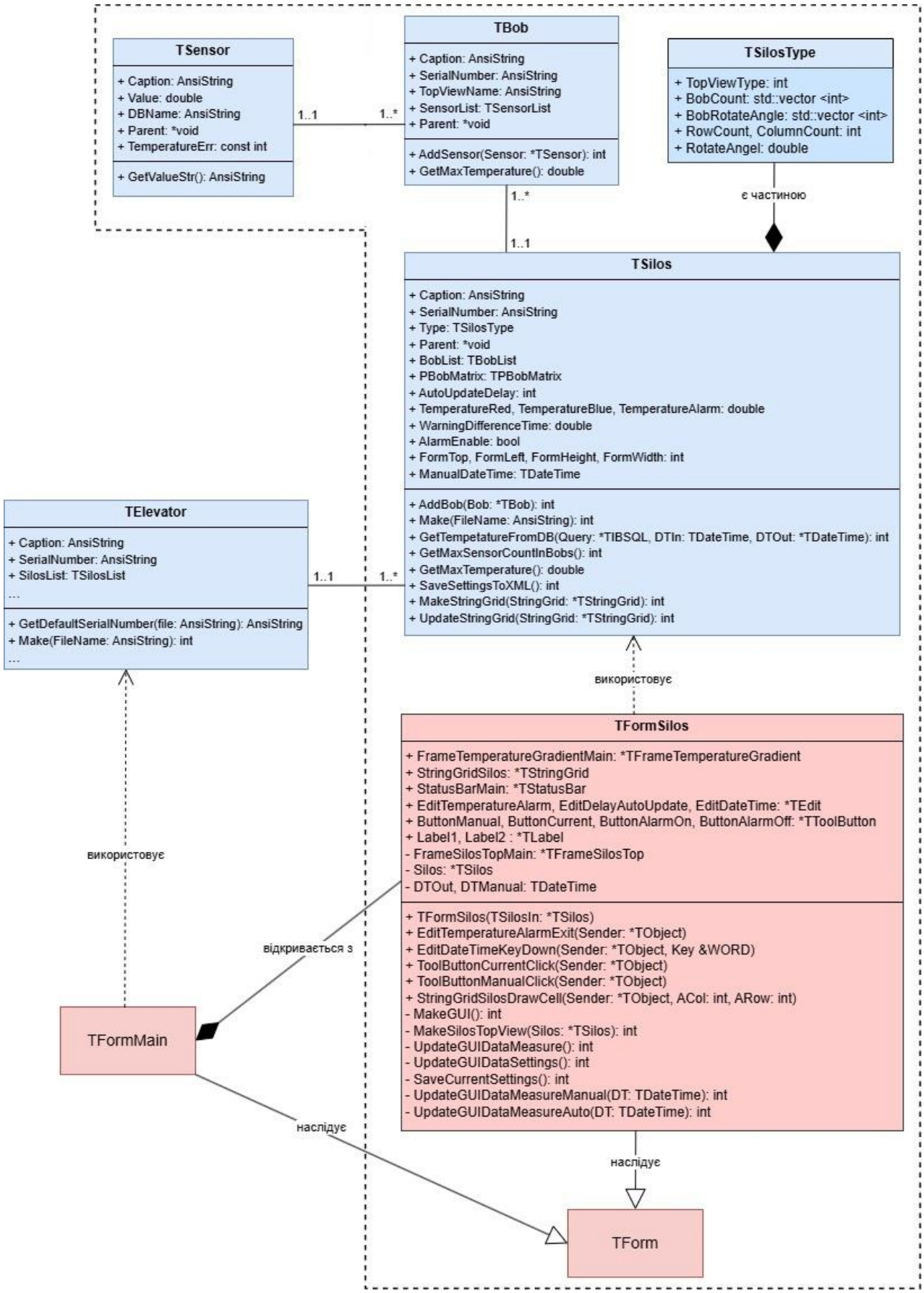

Рисунок 4.9 – UML-діаграма класів ПЗ Viewer

Клас Silos містить список підвісок BobList, що входять в нього, та матрицю вказівників на підвіски PBobMatrix для прямокутних силосів. Серед інших властивостей класу значення параметрів користувача форми TFormSilos.

Методи інформаційних класів можна поділити на три групи.

а) Внутрішні:

1) Make() – заповнює класи логіки (поля класів Silos, Bob, Sensor) даними з XML-файлів конфігурації та користувацьких налаштувань;

2) SaveSettingsToXML() – зберігає зміни у налаштуваннях у XML-файл користувацьких налаштувань та заповнює цей файл значеннями атрибутів за замовчуванням, якщо ці атрибути видалені;

3) додавання дочірніх вузлів у список: AddSensor() для класу Bob, AddBob() для класу Silos;

4) GetMaxTemperature() – визначає температуру на рівні певного інформаційного класу;

5) GetMaxSensorCountInBobs() – визначає максимальну кількість датчиків у підвісках.

б) Метод роботи з базою даних – GetTemperatureFromDB().

в) Методи, що працюють, з елементами віконного інтерфейсу:

1) MakeStringGrid() – формує таблицю у вікні TFormSilos, та заповнює заголовки стовпців та рядків іменами датчиків та підвісок;

2) UpdateStringGrid() с заповнює таблицю у вікні TFormSilos актуальними значеннями температури з властивості Value класу Sensor.

Метод GetTemperatureFromDB() класу Silos, який викликається в обробниках подій відкриття та оновлення форми TFormSilos, виконує підключення до БД, формування текстового рядка запиту, виконання транзакції та відключення від БД. Метод приймає три параметри: TIBSQL *Query – посилання на вбудований елемент C++Builder для роботи з БД; TDateTime DateTimeIn – дата та час, на який має відобразитися стан (вводиться

користувачем); TDateTime *DateTimeOut вказівник на час, найближчий знайдений у БД відносно DateTimeIn.

Таблицю TABLE_MEASURE, де зберігаються показники температури за часом наведено на рис. 4.10. Поле ID містить індекси рядків, DTLINECREATE – час та дату вимірювання, поля SENSOR0001…SENSORnnnn – показники температури з усіх датчиків в рамках силосу.

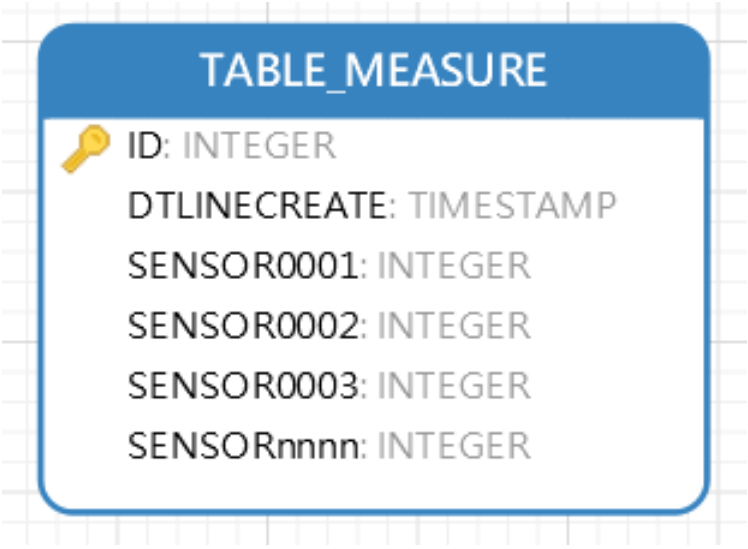

Рисунок 4.10 – Таблиця бази даних

Запит, який виконується до таблиці БД (лістинг 4.4), є динамічним, оскільки містить параметр P_DTLINECREATE, який отримується як вхідний аргумент DateTimeIn методу GetTemperatureFromDB().

```
SELECT FIRST 1 "SENSOR0001", "SENSOR0002", ..., "SENSORnnnn",
"DTLINECREATE", "ID"
FROM "TABLE_MEASURE"
WHERE DTLINECREATE<=:P_DTLINECREATE
ORDER BY "DTLINECREATE" DESC
```

Лістинг 4.4 – SQL-запит для отримання показників температури з датчиків за вхідними датою та часом

До того ж кількість полів у вибірці оператора SELECT, що представляють температуру датчиків (SENSOR0001…SENSORnnnn) буде залежати від конкретного силосу. Для цього в програмі виконується цикл по всім підвіскам силосу, що збирає імена усіх датчиків з властивості DBName класу Sensor.

Фактично запит підбирає рядок з датою, яка є найближчою до введеної користувачем, на що вказує ключове слово `FIRST` та конструкція сортування `ORDER BY "DTLINECREATE" DESC`. Дата та час підібрані за допомогою SQL-запиту будуть записані у змінну DateTimeOut.

4.2.3 Розробка розширення графічного інтерфейсу

Задачею цього розділу роботи є розробка додаткових елементів візуалізації для ПЗ Viewer, що дозволить просторово позначати температуру кожного датчика у силосах та розфарбовувати її за допомогою температурного градієнту, а також здійснювати користувацькі налаштування на рівні силосу.

Результатом мають стати додаткові вікна програми, представлені класом TFormSilos. Методи цього класу можна поділити на декілька груп:

– обробники подій елементів віконного інтерфейсу;

– методи спрямовані на роботу з елементами, що не є стандартними у середовищі розробки, але використовуються в рамках задачі (TFrameSilosTop, TFrameTemperatureGradient);

– допоміжні методи.

Вікно графічного інтерфейсу, що розроблено, показано на рис. 4.11.

Нестандартними елементами візуалізації та інтерфейсу, які розроблені для програми, але поза рамками цієї роботи є TFrameSilosTop та TFrameTemperatureGradient. Властивості та методи цих класів визначені у окремих файлах.

TFrameTemperatureGradient – клас, що представляє елемент інтерфейсу для виконання функцій температурного градієнту.

Клас TFrameSilosTop, що описує представлення круглих та прямокутних зерносховищ залежно від максимальної температури у підвісках. Цей клас містить метод UpdateGUI(), що перемальовує представлення залежно від показників температури за допомогою температурного градієнта.

Вікно (форма), що представлене класом TSilosForm, має стандартні для середи розробки події:

а) відкриття форми (OnShow);

б) малювання форми (OnPaint);

в) закриття форми (OnClose).

Важливим методом також є конструктор форми.

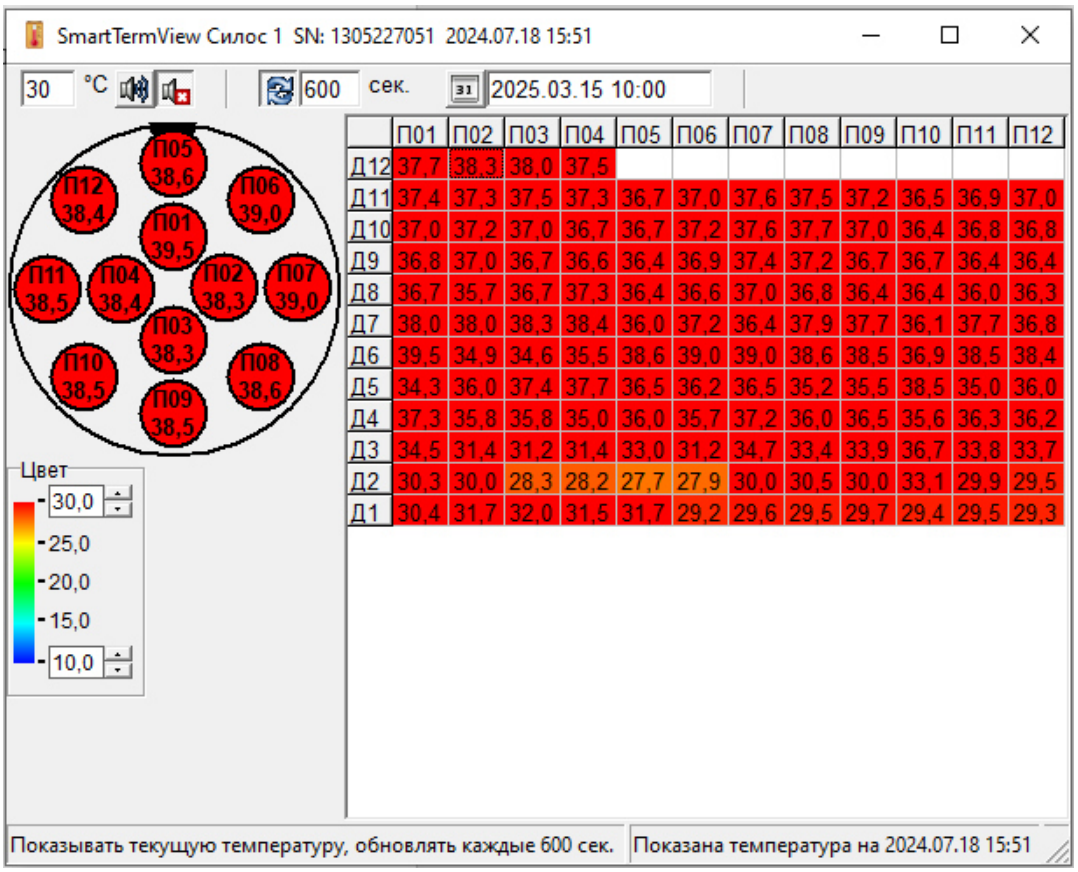

Рисунок 4.11 – Інтерфейс вікна, що розроблене, під час роботи в автоматичному режимі

Створення форми відбувається у класі головної форми TFormMainSilos, яка вже наявна у проекті, один раз при запуску програми. Надалі форма TFormSilos лише відкривається при виборі відповідного зерносховища у вкладці «Сховища».

У конструктор об’єкту TFormSilos передаються три параметри:

– TComponent* Owner = Application;

– TSilos *SilosIn – посилання на об’єкт типу TSilos, що відповідає зерносховищу, яке візуалізує це вікно;

– TIBDatabase *IBDatabase – посилання на вбудований елемент для роботи з базою даних.

До того ж у конструкторі викликається метод MakeSilosTopView(), який визначає положення елементів типу TFrameSilosTop, TFrameTemperatureGradient, TStringGrid у вікні, що розробляється.

Подія OnShow відбувається при відкритті форми, та обробляється методом FormShow (додаток Б). Метод FormShow викликає інші методи, які встановлюють властивості вікна та елементів інтерфейсу у початковий стан згідно зі значеннями відповідних властивостей класу TSilos. Також обробник FormShow викликає методи GetTemperatureFromDB() та UpdateGUIDataMeasure() для звернення до бази даних за показниками температури та оновлення візуалізації стану зерносховища на поточний час. Вмикається таймер звукового сигналу, та встановлюється автоматичний режим.

Перелік методів, що викликаються у обробнику FormShow:

– MakeGUI() – визначає положення форми, будує таблицю із заповненими заголовками, робить кнопку дозволу звукового сигналу натиснутою, якщо це відповідає налаштуванням;

– UpdateGUIDataSettings() – встановлює у текстові поля вводу, що відповідають параметрам користувача, значення із класу TSilos та задає натиснення кнопок;

– GetTemperatureFromDB() – виконує звернення до бази даних на поточний час;

– UpdateGUIDataMeasure() – оновлює візуалізацію температурного стану згідно з поточними даними;

– ToolButtonCurrentClick() – запускає таймер автоматичного оновлення та встановлює його інтервал, робить кнопку оновлення натиснутою.

Наступна подія OnPaint відбувається багаторазово при необхідності перемалювати графічний інтерфейс вікна (зміна розміру вікна, перемикання між вікнами, тощо). Подію OnPaint обробляє метод FormPaint, призначенням якого є виведення попередження про застарілі дані у базі даних за допомогою діалогового вікна типу MessageBox. Це діалогове вікно повідомляє про кількість годин на яку застарілі дані у БД. Для концентрації оператора на роботі діалогове вікно спливає один раз для кожного зерноскладу, за що відповідає статична змінна HadPainted.

Подію закриття вікна TFormSilos обробляє метод FormClose, який зупиняє обидва таймера та виводить діалогове вікно з пропозицією зберегти зміни у користувацьких налаштуваннях, якщо вони були зроблені. Метод SaveCurrentSettings(), що викликається методом-обробником, копіює значення властивостей класу TFormSilos у відповідні властивості класу TSilos, а звідти вже у файл STViewer2000v205_US.XML. Діаграма, що показує за допомогою яких методів значення користувацьких налаштувань зчитуються/записуються у класи та XML-файл, наведена на рис. 4.12.

Оскільки форма TFormSilos має працювати у ручному та автоматичному режимах, то для вирішення цієї задачі у панелі управління розміщені дві групи елементів: кнопки; поля для введення дати та часу, періоду звернення до БД. Взаємодія з цими елементами інтерфейсу викликає обробники подій, що використовують у своєму тілі розроблені методи UpdateGUIDataMeasureAuto(), UpdateGUIDataMeasureManual() для оновлення візуалізації при автоматичному та ручному режимах відповідно. Ці два методи відрізняються інформацією, що виводиться у рядок стану (рис. 4.13).

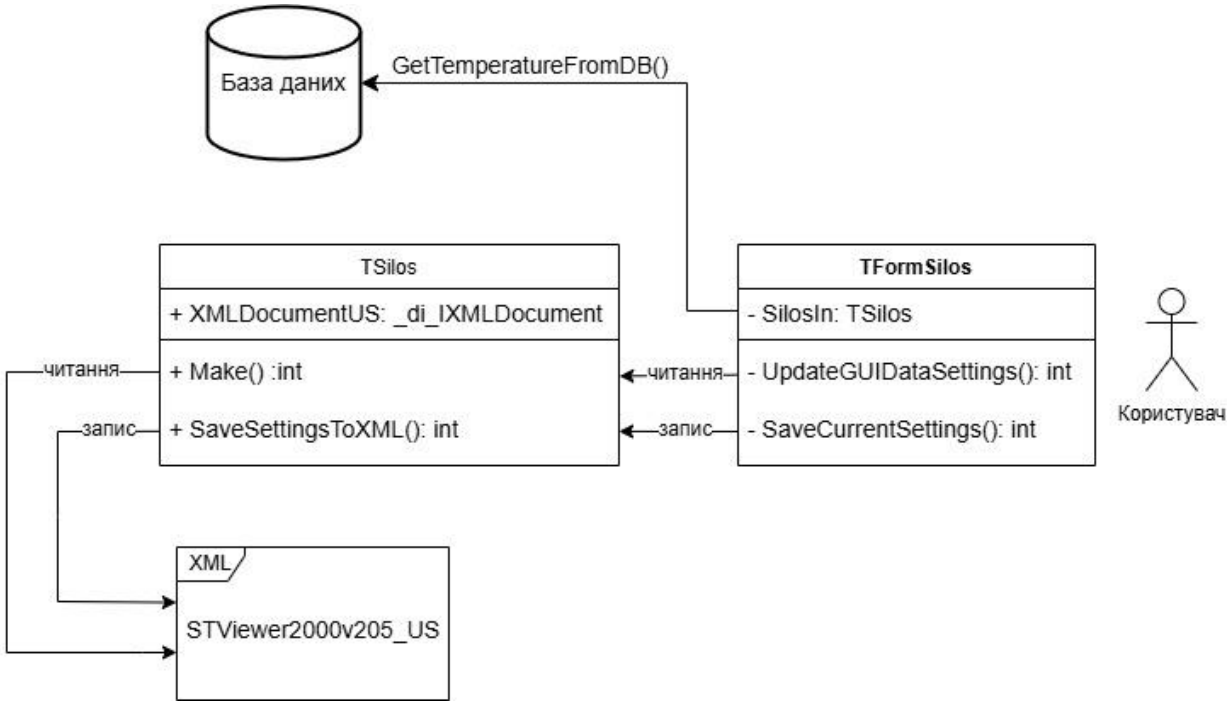

Рисунок 4.12 – Запис та читання налаштувань користувача і звернення до БД

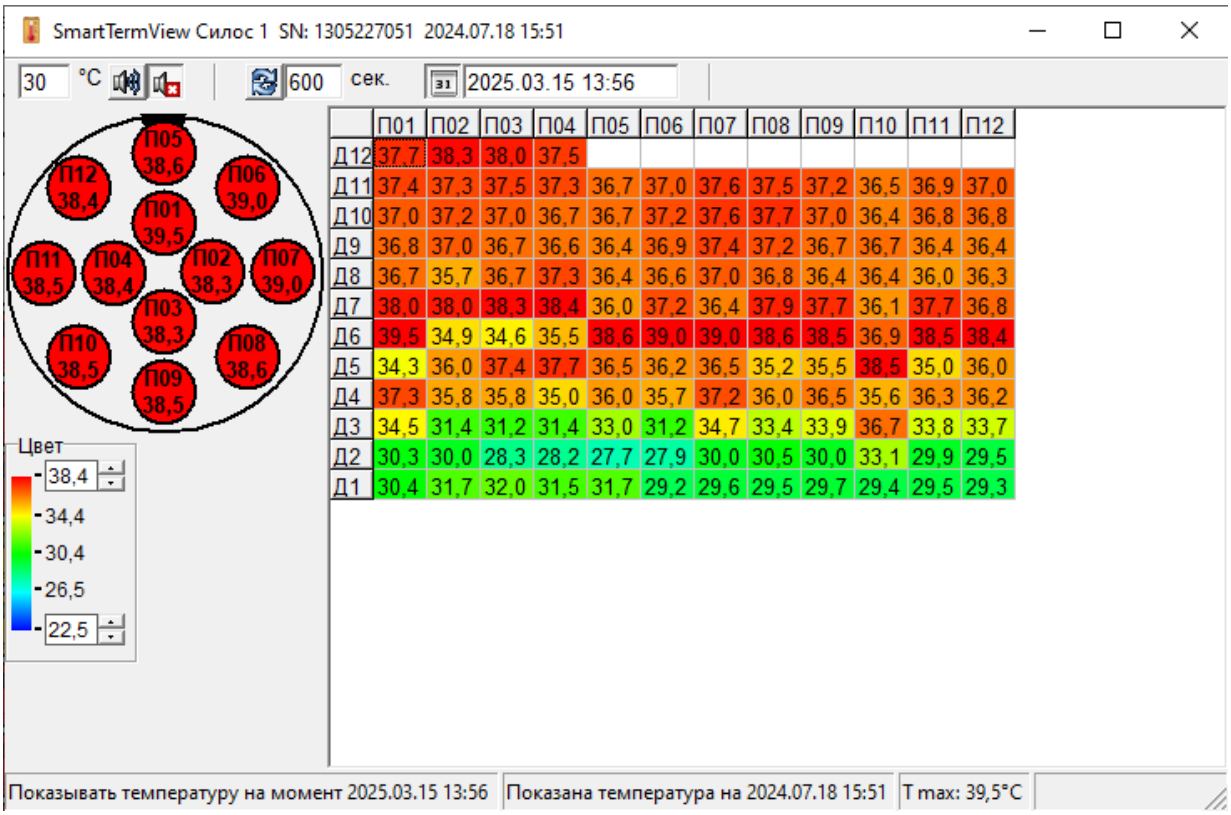

Рисунок 4.13 – Вигляд вікна, що розроблене, при роботі в ручному режимі

Вікно TFormSilos включає два прихованих елементи управління:

– таймер звукового сигналу: кожну секунду викликає обробник, який відтворює звуковий сигнал, якщо перейдено межу допустимої температури;

– таймер автоматичного оновлення: працює по інтервалам, які зчитуються з редагованого текстового поля періоду автоматичного оновлення. В обробнику події цього таймера викликається метод UpdateGUIDataMeasureAuto().

До методів-обробників подій елементів інтерфейсу, що пов’язані з автоматичним режимом відносяться наступні.

а) ToolButtonCurrentClick() – спрацьовує при натисненні кнопки оновлення, миттєво оновлює температурний стан:

1) відключає кнопку ручного оновлення температури;

2) встановлює новий інтервал для таймера автоматичного оновлення, ніби відбулося введення;

3) викликає обробник спрацювання таймера автоматичного оновлення, ніби інтервал пройшов.

б) EditDelayAutoUpdateExit() – спрацьовує при натисненні кнопок миші на нейтральній області вікна після введення у редаговане поле періоду оновлення, викликає метод EditDelayAutoUpdateKeyDown().

в) EditDelayAutoUpdateKeyDown() – викликається при натисканні клавіш Enter/Esc після введення у редаговане текстове поле періоду оновлення, оновлює інтервал спрацювання таймера, використовується в обох попередніх методах-обробниках.

г) AutoUpdateTimer() – обробник таймера автоматичного оновлення, викликає метод UpdateGUIDataMeasureAuto().

Методи-обробники подій елементів інтерфейсу, які пов'язані з ручним режимом роботи є наступними:

– ToolButtonManualClick() – спрацьовує при натисканні кнопки ручного оновлення температури: вимикає кнопку автоматичного оновлення, викликається обробник події ніби вводилися дані у текстове поле дати та часу;

– EditDateTimeExit() – спрацьовує при натисненні кнопок миші на нейтральній області вікна після введення у редаговане поле дати та часу;

– EditDateTimeKeyDown() – викликається при натисканні клавіш Enter/Esc після введення у редаговане текстове поле дати та часу, викликає метод оновлення температурного стану UpdateGUIDataMeasureManual(), використовується в обох попередніх методах-обробниках.

Серед інших методів-обробників форми:

– EditTemperatureAlarmExit() – викликається після натискання кнопок миші на нейтральній області вікна після введення у текстове поле межі допустимої температури;

– AlarmTimerHandler() – обробник події таймера звукового сигналу, якщо максимальна температура у зерносховищі більша, ніж допустима – відтворюється сигнал за допомогою вбудованої функції Beep();

– StringGridSilosDrawCell() – змінює колір комірки таблиці за допомогою температурного градієнта, якщо змінився текст у комірці.

ВИСНОВКИ

Представлену дипломну роботу спрямовано на проектування і реалізацію елементів апаратно-програмного комплексу моніторингу температурного режиму зерносховищ.

Розглянуто елементи предметної області систем термометрії зерносховищ з моніторингом серед яких: протокол передачі даних 1-Wire для підключення великої кількості цифрових датчиків температури DS18B20, структура апаратних та програмних елементів температурного моніторингу елеваторів, функціональна схема пристрою адаптації аналогових термопідвісок до сучасного комплексу (модуля 1W-Rx6), процедура завантаження бінарного файлу програми у мікроконтролер (прошивання), процедура обробки та візуалізації температурних даних.

Розроблено програмне забезпечення для модуля 1W-Rx6 на базі мікроконтролера ATtiny44, який адаптує термопідвіски з аналоговими датчиками до наявного комплексу моніторнигу температури шляхом емуляції цифрових датчиків DS18B20 та передачі даних по шині 1-Wire. Програма має поширену для прошивок мікроконтролерів структуру: відокремлені тіло програми та обробники апаратних переривань. Створено і описано загальну блок-схему програми та блок-схеми окремих її частин. В якості середовища розробки прошивки використовувалося Atmel Studio 7.0, а для прошивання програматор-адаптер на мікросхемі FT232HQ.

Для покращення сприйняття даних на етапі візуалізації та розширення функціоналу для ПЗ Viewer розроблено дочірнє вікно, яке шляхом використання елементів інтерфейсу схематично позначає температуру кожного датчика у просторі зерносховища, а також дозволяє користувачу встановлювати параметри моніторингу окремо для кожного зерносховища. У ході вирішення задачі розробки розширених елементів візуалізації для Viewer розглядалися та використовувалися деякі вже наявні в програмі класи. В якості

середовища розробки для розширення функціональності Viewer використовувалося C++Builder, а в якості СУБД Firebird.

Розробка розширення візуалізації для ПЗ Viewer спрямована на підвищення ефективності моніторингу за температурним станом на кожному зерносховищі та елеваторі загалом. Водночас розробка прошивки для модуля 1W-Rx6 дозволяє цьому модулю адаптувати до використання старі, та ще наявні на елеваторах термопідвіски з аналоговими датчиками, здешевшуючи повну або часткову модернізацію інших комплексів моніторингу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Круш А.І., Малахов Є.В. Реалізація деяких елементів програмно-апаратного комплексу моніторингу температурного режиму зерносховищ // Інформатика, інформаційні системи та технології: тези доповідей XXII Всеукраїнської конференції студентів і молодих науковців. - Одеса, 25 квітня 2025 р. - Одеса, 2025. С. - 201-202.

2. А.О. Новацький Проектування мережі 1-WIRE: Навчальний посібник для студентів спеціальностей «Комп'ютеризовані системи управління та автоматика» кафедри автоматики та управління в технічних системах / К: НТУУ «КПІ», 2014 – 141с.

3. DS18B20 - Programmable Resolution 1-Wire Digital Thermometer [Електронний ресурс] – Режим доступу: https://www.analog.com/media/en/ technical-documentation/data-sheets/DS18B20.pdf - Дата звернення: 29.01.2025

4. Гунченко Ю.О. Інтелектуальні засоби вимірювань: однокристальні мікроконтролери AVR (навчальний посібник). Частина 1. Архітектура, система команд, порти вводу/виводу, переривання. – Одеса: ВМВ, 2011. 184 с. (гриф МОНУ. Лист № 1.4/18-Г-77 від 10.01.2009).

5. ATtiny24/44/84 [Електронний ресурс] – Режим доступу: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7701_Automotive-Microcontrollers-ATtiny24-44-84_Datasheet.pdf - Дата звернення: 29.01.2025

6. 8-channel analog multiplexer/demultiplexer [Електронний ресурс] – Режим доступу: https://assets.nexperia.com/documents/data-sheet/74HC_HCT4051.pdf - Дата звернення: 29.01.2025

7. Using Firebird: (work in progress) [Електронний ресурс] – Режим доступу: https://www.firebirdsql.org/file/documentation/html/en/firebirddocs/ufb/using-firebird.html#ufb-cs-embedded – Дата звернення: 10.02.2025

ДОДАТОК А

Конфігураційний файл користувацьких налаштувань STViewer2000v205_US.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SmartTerm>
<DataBase User="SYSDBA" Password="masterkey" Port="" Server=""
DataBaseName="SM2000_053" Role=""/>
<Object SerialNumber="1122" TemperatureRed="36,9"
TemperatureBlue="10" TemperatureAlarm="40" AlarmEnable="False"
AutoUpdateDelay="600" ManualDateTime="2024.05.15 13:56"
FormTop="64" FormLeft="62" FormHeight="762" FormWidth="939"
TreeShow="False" WarningDifferenceTime="72" ImageScale="100">
<Silos SerialNumber="1510297033" TemperatureRed="39,9"
TemperatureBlue="6,4" TemperatureAlarm="30" AlarmEnable="False"
AutoUpdateDelay="600" ManualDateTime="2024.02.15 13:56"
FormTop="49" FormLeft="0" FormHeight="890" FormWidth="1215"/>
<Silos SerialNumber="1305227051" TemperatureRed="38,4"
TemperatureBlue="20,1" TemperatureAlarm="30" AlarmEnable="False"
AutoUpdateDelay="600" ManualDateTime="2024.04.05 13:56"
FormTop="227" FormLeft="203" FormHeight="435" FormWidth="752"/>
<Silos SerialNumber="1305227052" TemperatureRed="39"
TemperatureBlue="10" TemperatureAlarm="40" AlarmEnable="True"
AutoUpdateDelay="600" ManualDateTime="2024.05.15 13:56"
FormTop="86" FormLeft="84" FormHeight="461" FormWidth="760"/>
</Object>
</SmartTerm>
```

Лістинг А.1 – Конфігураційний XML-файл користувацьких налаштувань

ДОДАТОК Б

Вихідний код класу TFormSilos програми Viewer

```
#include <vcl.h>
#include "STViewerFormSilos.h"
#pragma hdrstop
#pragma package(smart_init)
#pragma link "CSPIN"
#pragma link "STViewerTemperatureGradient"
#pragma resource "*.dfm"
TFormSilos *FormSilos;

__fastcall TFormSilos::TFormSilos(TComponent* Owner)
    : TForm(Owner)
{
    ShowFirst=true;
}
__fastcall TFormSilos::TFormSilos(TComponent* Owner, TSilos
*SilosIn,TIBDatabase *IBDatabase)
    : TForm(Owner)
{
    this->IBSQLTemp->Database=IBDatabase;
    this->TTemp->DefaultDatabase=IBDatabase;
    this->Silos=SilosIn;
    this->MakeSilosTopView(SilosIn);

    ShowFirst=true;
}
int TFormSilos::MakeSilosTopView(TSilos *Silos)
{

    FrameSilosTopMain = new TFrameSilosTop(this, this, Silos);
    FrameSilosTopMain->Parent=this;
    FrameSilosTopMain->Left=0;
    FrameSilosTopMain->Top=34;
    FrameSilosTopMain->Name="FrameSilosTopMain";

 FrameTemperatureGradientMain->Top=34+FrameSilosTopMain->Height;
    if (FrameTemperatureGradientMain->Width<FrameSilosTopMain-
>Width) StringGridSilos->Width=ClientWidth-(2+FrameSilosTopMain-
>Width);
    else StringGridSilos->Width=ClientWidth-
(2+FrameTemperatureGradientMain->Width);

    return 0;
}
```

Лістинг Б.1 – Вихідний код для створення дочірніх вікон ПЗ Viewer, що розроблені

```
int TFormSilos::SaveCurrentSettings()
{
    Silos->TemperatureRed   = FrameTemperatureGradientMain-
>TemperatureRed;
    Silos->TemperatureBlue  = FrameTemperatureGradientMain-
>TemperatureBlue;
    Silos->TemperatureAlarm = EditTemperatureAlarm-
>Text.ToDouble();
    Silos->AlarmEnable      = ToolButtonAlarmOn->Down;
    Silos->AutoUpdateDelay  = EditDelayAutoUpdate-
>Text.ToIntDef(Silos->AutoUpdateDelay);
    Silos->ManualDateTime   = DTManual;

    Silos->FormTop=Top;
    Silos->FormLeft=Left;
    Silos->FormHeight=Height;
    Silos->FormWidth=Width;
    return Silos->SaveSettingsToXML();
}
void __fastcall TFormSilos::StringGridSilosDrawCell(TObject
*Sender,
      int ACol, int ARow, TRect &Rect, TGridDrawState State)
{
    switch (Silos->Type.TopViewType) {
      case Silos->Type.Round :{
        if ((ACol == 0) || (ARow == 0))    return;
        break;
        }
      case Silos->Type.Square :{
        if ((ACol == 0) /*|| (ARow == 0)*/)    return;
        break;
        }
      }

    TStringGrid *sg = (TStringGrid*)Sender;
    AnsiString Str = sg->Cells[ACol][ARow];

    if (((&Str)!=NULL)&&(Str!="")&&((Str.LowerCase()!="err"))){
        float TempValue=StrToFloatDef(Str,-10000);
        if (-1000<TempValue) sg->Canvas->Brush-
>Color=SmartTermUtil::ValueToColor(FrameTemperatureGradientMain-
>TemperatureRed,FrameTemperatureGradientMain-
>TemperatureBlue,TempValue);
          else  sg->Canvas->Brush->Color=clWhite;
        }
    if (0<Str.Pos(" ")) sg->Canvas->Brush->Color=clBtnFace;
  sg->Canvas->FillRect(Rect);
  sg->Canvas->TextRect(Rect, Rect.Left + 2, Rect.Top + 2, Str);
}
```

Лістинг Б.1, аркуш 2

```
int TFormSilos::UpdateGUIDataSettings()
{
    this->Caption="SmartTermView "+Silos->Caption+"  SN:
"+Silos->SerialNumber;
    EditDelayAutoUpdate->Text=IntToStr(Silos->AutoUpdateDelay);
    EditDateTime->Text=Silos->ManualDateTime.DateTimeString();
    DTManual=Silos->ManualDateTime;
    FrameTemperatureGradientMain->TemperatureRed=Silos-
>TemperatureRed;
    FrameTemperatureGradientMain->TemperatureBlue=Silos-
>TemperatureBlue;

    EditTemperatureAlarm->Text= FormatFloat("0.#",Silos-
>TemperatureAlarm);
    ToolButtonAlarmOff->Down=!Silos->AlarmEnable;
    ToolButtonAlarmOn->Down=Silos->AlarmEnable;
    return 0;
}
int TFormSilos::UpdateGUIDataMeasure()
{
    this->Caption="SmartTermView "+Silos->Caption+"  SN:
"+Silos->SerialNumber+"  "+ DTOut.DateTimeString();
    Silos->UpdateStringGrid(StringGridSilos);
    FrameSilosTopMain->UpdateGUI(FrameTemperatureGradientMain-
>TemperatureRed,FrameTemperatureGradientMain->TemperatureBlue);

    double MaxT=Silos->GetMaxTemperature();
    if (MaxT!=TSensor::TemperatureErr) StatusBarMain->Panels-
>Items[2]->Text="T max: "+FormatFloat("0.0",MaxT)+"°C";
      else StatusBarMain->Panels->Items[2]->Text="T max: Err";
    StatusBarMain->Canvas->Font=StatusBarMain->Font;
    StatusBarMain->Panels->Items[2]->Width=9+StatusBarMain-
>Canvas->TextWidth(StatusBarMain->Panels->Items[2]->Text);
    return 0;
}
void __fastcall TFormSilos::AutoUpdateTimer(TObject *Sender)
{
    AutoModeTimer->Enabled=false;
    if (ToolButtonCurrent->Down){
        TDateTime DTNow=Now();
        UpdateGUIDataMeasureAuto(DTNow);
        AutoModeTimer->Enabled=ToolButtonCurrent->Down;
        }
}
void __fastcall TFormSilos::ToolButtonCurrentClick(TObject
*Sender)
{
    if (!ToolButtonManual->Down) {ToolButtonCurrent->Down=true;}
    if(ToolButtonCurrent->Down) {
```

Лістинг Б.1, аркуш 3

```
        ToolButtonManual->Down=false;
        unsigned short Key=VK_RETURN;
        EditDelayAutoUpdateKeyDown(NULL,Key,TShiftState());
        AutoUpdateTimer(NULL);
        AutoModeTimer->Enabled=ToolButtonCurrent->Down;
        }
}
void __fastcall TFormSilos::ToolButtonManualClick(TObject
*Sender)
{
    if (!ToolButtonCurrent->Down) {ToolButtonManual->Down=true;}
    if (ToolButtonManual->Down) {
        ToolButtonCurrent->Down=false;
        unsigned short Key=VK_RETURN;
        EditDateTimeKeyDown(Sender, Key, TShiftState());
        }
}
void __fastcall TFormSilos::EditDelayAutoUpdateExit(TObject
*Sender)
{
    unsigned short Key=VK_RETURN;
    EditDelayAutoUpdateKeyDown(NULL,Key,TShiftState());
}
void __fastcall TFormSilos::EditTemperatureAlarmExit(TObject
*Sender)
{
    double i;
    try {
        i=StrToFloat(EditTemperatureAlarm->Text.Trim());
        }
     catch(...) {
        Beep(1000,100);
        i=Silos->TemperatureAlarm;
        EditTemperatureAlarm->SetFocus();
        }
    EditTemperatureAlarm->Text=FormatFloat("0.#",i);
}
void __fastcall TFormSilos::EditDateTimeExit(TObject *Sender)
{
    unsigned short Key=VK_RETURN;
    EditDateTimeKeyDown(NULL, Key, TShiftState());
}
void __fastcall TFormSilos::EditDateTimeKeyDown(TObject *Sender,
WORD &Key,
      TShiftState Shift)
{
    DateSeparator='.';
    TimeSeparator=':';
    ShortDateFormat="yyyy.mm.dd";
```

Лістинг Б.1, аркуш 4

```
    ShortTimeFormat="hh:nn";
    LongTimeFormat="hh:nn";
    if(Key==VK_RETURN)
        try{
            DTManual=StrToDateTime (EditDateTime->Text.Trim());
            }
          catch(...) {
            Application->MessageBoxA(("Не правильно введені дата
та час"
                                      "\r\nПриклад дати та часу:
"+Now().DateTimeString()+
                                      "\r\nВведіть дату та час ще
раз.").c_str(),"Помилка введення
даних!",MB_OK|MB_ICONERROR|MB_DEFBUTTON1);
            EditDateTime->SetFocus();
            }
    if (Key==VK_ESCAPE) EditDateTime-
>Text=DTManual.DateTimeString();

    if ((Key==VK_ESCAPE)||(Key==VK_RETURN)){
        if (ToolButtonManual->Down)
UpdateGUIDataMeasureManual(DTManual);
        }
}
void __fastcall TFormSilos::FormCreate(TObject *Sender)
{
    DateSeparator='.';
    TimeSeparator=':';
    ShortDateFormat="yyyy.mm.dd";
    ShortTimeFormat="hh:nn";
    LongTimeFormat="hh:nn";
}
void __fastcall TFormSilos::FormShow(TObject *Sender)
{
    if (ShowFirst) {
        MakeGUI();
        UpdateGUIDataSettings();
        Silos->GetTempetatureFromDB(IBSQLTemp,Now(),&DTOut);
        UpdateGUIDataMeasure();
        ToolButtonCurrentClick(NULL);
        AlarmTimer->Enabled=true;
    }
}
int TFormSilos::MakeGUI()
{
    Top=Silos->FormTop;
    Left=Silos->FormLeft;
    Height=Silos->FormHeight;
    Width=Silos->FormWidth;
```

Лістинг Б.1, аркуш 5

```
    if (Silos->AlarmEnable) ToolButtonAlarmOn->Down=true;
      else ToolButtonAlarmOff->Down=true;

    Silos->MakeStringGrid(StringGridSilos);

    return 0;
}
void __fastcall TFormSilos::FormClose(TObject *Sender,
      TCloseAction &Action)
{
    AutoModeTimer->Enabled=false;
    AlarmTimer->Enabled=false;

    bool IsEqual=true;
    IsEqual = IsEqual&&(Silos-
>TemperatureRed==FrameTemperatureGradientMain->TemperatureRed);
    IsEqual = IsEqual&&(Silos-
>TemperatureBlue==FrameTemperatureGradientMain-
>TemperatureBlue);
    IsEqual = IsEqual&&(0.1>fabs(Silos->TemperatureAlarm-
(EditTemperatureAlarm->Text.ToDouble())));
    IsEqual = IsEqual&&(Silos->AlarmEnable==ToolButtonAlarmOn-
>Down);
    IsEqual = IsEqual&&(Silos-
>AutoUpdateDelay==EditDelayAutoUpdate-
>Text.Trim().ToIntDef(60));
    IsEqual = IsEqual&&(Silos->ManualDateTime==DTManual);
    IsEqual = IsEqual&&(Silos->FormTop==Top);
    IsEqual = IsEqual&&(Silos->FormLeft==Left);
    IsEqual = IsEqual&&(Silos->FormHeight==Height);
    IsEqual = IsEqual&&(Silos->FormWidth==Width);

    if (!IsEqual)
        if (IDYES==Application->MessageBoxA("При роботі були
змінені налаштування програми."
                                    "\r\nДля того, щоб зберігти
нові налаштування натисніть ТАК"
                                    "\r\nДля того, щоб залишити
старі налаштування натисніть НІ","Збереження
налаштувань.",MB_YESNO|MB_ICONQUESTION|MB_DEFBUTTON1))
            SaveCurrentSettings();
}
void __fastcall
TFormSilos::FrameTemperatureGradientMainEditTemperatureRedChange
(
      TObject *Sender)
{
    UpdateGUIDataMeasure();
}
```

Лістинг Б.1, аркуш 6

```
void __fastcall
TFormSilos::FrameTemperatureGradientMainEditTemperatureBlueChang
e(
      TObject *Sender)
{
    UpdateGUIDataMeasure();
}
void __fastcall
TFormSilos::FrameTemperatureGradientMainUpDownRedChanging(
      TObject *Sender, bool &AllowChange)
{
    UpdateGUIDataMeasure();
}

void __fastcall
TFormSilos::FrameTemperatureGradientMainUpDownBlueChanging(
      TObject *Sender, bool &AllowChange)
{
    UpdateGUIDataMeasure();
}

void __fastcall TFormSilos::AlarmTimerHandler(TObject *Sender)
{
    AlarmTimer->Enabled=false;
    if (ToolButtonAlarmOn->Down) {
        double AlarmT;
        try {
            AlarmT=StrToFloat(EditTemperatureAlarm-
>Text.Trim());
            }
         catch(...) {
            Beep(1000,100);
            AlarmT=Silos->TemperatureAlarm;
            EditTemperatureAlarm->SetFocus();
            }
        EditTemperatureAlarm->Text=FormatFloat("0.#",AlarmT);
            if (AlarmT<Silos->GetMaxTemperature())
{Beep(1500,150);}
        }
    AlarmTimer->Enabled=true;
}

int TFormSilos::UpdateGUIDataMeasureManual(TDateTime DT)
{
    StatusBarMain->Canvas->Font=StatusBarMain->Font;
    StatusBarMain->Panels->Items[0]->Text="Показывать
температуру на момент "+DT.DateTimeString();
    StatusBarMain->Panels->Items[0]->Width=9+StatusBarMain-
>Canvas->TextWidth(StatusBarMain->Panels->Items[0]->Text);
```

Лістинг Б.1, аркуш 7

```
    Silos->GetTempetatureFromDB(IBSQLTemp,DT,&DTOut);
    UpdateGUIDataMeasure();
    if (DTOut.Val!=0) StatusBarMain->Panels->Items[1]-
>Text="Показана температура на "+DTOut.DateTimeString();
        else StatusBarMain->Panels->Items[1]->Text="Измерение на
указанную дату "+DT.DateTimeString()+" не найдено";
    StatusBarMain->Panels->Items[1]->Width=9+StatusBarMain-
>Canvas->TextWidth(StatusBarMain->Panels->Items[1]->Text);
    return 0;
}

int TFormSilos::UpdateGUIDataMeasureAuto(TDateTime DT)
{
    StatusBarMain->Canvas->Font=StatusBarMain->Font;
    StatusBarMain->Panels->Items[0]->Text="Показывать текущую
температуру, обновлять каждые "+EditDelayAutoUpdate-
>Text.Trim()+" сек.";
    StatusBarMain->Panels->Items[0]->Width=9+StatusBarMain-
>Canvas->TextWidth(StatusBarMain->Panels->Items[0]->Text);
    Silos->GetTempetatureFromDB(IBSQLTemp,DT,&DTOut);
    UpdateGUIDataMeasure();
    if (DTOut.Val!=0) StatusBarMain->Panels->Items[1]-
>Text="Показана температура на "+DTOut.DateTimeString();
        else StatusBarMain->Panels->Items[1]->Text="Измерение на
указанную дату "+DT.DateTimeString()+" не найдено";
    StatusBarMain->Panels->Items[1]->Width=9+StatusBarMain-
>Canvas->TextWidth(StatusBarMain->Panels->Items[1]->Text);
    return 0;
}

void __fastcall TFormSilos::EditDelayAutoUpdateKeyDown(TObject
*Sender,
      WORD &Key, TShiftState Shift)
{
    int i;
    bool BoolTemp=AutoModeTimer->Enabled;
    AutoModeTimer->Enabled=false;
    if(Key==VK_RETURN)
        try{
            i=StrToInt(EditDelayAutoUpdate->Text.Trim());;
            }
          catch(...) {
            Beep(1000,100);
            i=Silos->AutoUpdateDelay;
            EditDelayAutoUpdate->SetFocus();
            }

    if(Key==VK_ESCAPE) i=Silos->AutoUpdateDelay;
```

Лістинг Б.1, аркуш 8

```
    if((Key==VK_RETURN)||(Key==VK_ESCAPE)) {

        EditDelayAutoUpdate->Text=IntToStr(i);
        AutoModeTimer->Interval=1+(i*1000);
        }
    AutoModeTimer->Enabled=BoolTemp;
}

void __fastcall TFormSilos::FormPaint(TObject *Sender)
{
    static bool HadPainted=false;
    if (!HadPainted) {

        if ( ToolButtonCurrent->Down&&(Silos-
>WarningDifferenceTime<24*(Now().Val-DTOut.Val)))
           Application->MessageBoxA(("Дані застаріли на
"+IntToStr((int)(24*(Now().Val-DTOut.Val)))+" годин."
                                     "\r\nДата останнього
вимірювання: "+DTOut.DateTimeString() +
                                     "\r\nПоточна дата:
"+Now().DateTimeString() +
                                     "\r\nПеревірте
працездатність програми Registrator,"
                                     "\r\nПеревірте справність
обладнання,"
                                     "\r\nПеревірте правильність
встановленої дати та часу на комп’ютері,"
                                     "\r\nЗверніться у службу
технічної підтримки.").c_str()
                                     ,"Дані
застаріли!",MB_OK|MB_ICONWARNING|MB_DEFBUTTON1);
        HadPainted=true;
        }
}
```

Лістинг Б.1, аркуш 9

ДОДАТОК В

Тези доповідей на XXII Всеукраїнську конференцію студентів та молодих науковців

РЕАЛІЗАЦІЯ ДЕЯКИХ ЕЛЕМЕНТІВ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ МОНІТОРИНГУ ТЕМПЕРАТУРНОГО РЕЖИМУ ЗЕРНОСХОВИЩА

Круш А.І., Малахов Є.В.

Одеський національний університет імені І.І. Мечникова

Ключові слова: мікроконтролер, датчики температури, шина 1-Wire .

Процеси зберігання та сушіння агропродукції у спеціалізованих сховищах вимагають моніторингу температури у різних точках. Сучасні програмно-апаратні системи моніторингу можуть бути реалізовані на основі цифрових датчиків температури DS18B20 [1]. Перевагою цієї моделі датчиків є підтримка передачі даних двонаправленою шиною 1-Wire, яка дозволяє підключити велику їх кількість до одного мікроконтролера. Надалі показники температури зчитуються із пам'яті мікроконтролера комп'ютером верхнього рівня для подальшої обробки.

Проте при модернізації старих зерносховищ виникає потреба використання раніше встановлених старих термопідвісок із аналоговими датчиками температури. Робота присвячена розробці програмного забезпечення для спеціалізованого апаратного модуля 1W-Rx6, який при підключених аналогових датчиках температури, здатен забезпечити повну емуляцію підвіски з цифрових датчиків DS18B20 на шині 1-Wire (рис. 1). Модуль отримав таку назву через програмну реалізацію шини 1-Wire та 6 терморезисторів, що підключаються до нього.

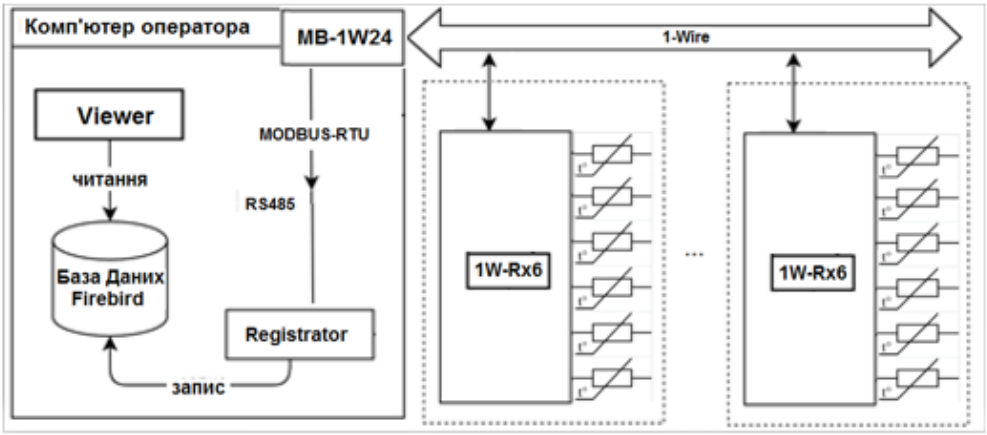

Рисунок 1.1 – Структура елементів апаратно-програмного комплексу

Рисунок В.1 – Тези доповідей на XXII конференцію, аркуш 1

Модуль 1W-Rx6 побудований на базі 8-бітного AVR-мікроконтролера ATtiny44 [2]. Для зчитування температури з аналогових датчиків використовується 10-бітовий вбудований канал АЦП. До шини 1-Wire мікроконтролер підключається лініями вводу-виводу загального призначення, а реагування на зміни стану шини відбувається за допомогою апаратних переривань. В рамках роботи використовується значна частина внутрішніх пристроїв мікроконтролера.

Програмне забезпечення (прошивка) для мікроконтролера ATtiny44 написана мовою Асемблера в інтегрованому середовищі розробки (IDE) Atmel Studio 7.0 від компанії Atmel. Програмно реалізовано як протокол обміну даними по шині 1-Wire, так і інтерфейс, що емулює взаємодію з цифровими датчиками DS18B20. Температурні показники, що передаються, зберігаються в БД на комп'ютері верхнього рівня. Для цього використовується реляційна SQL-серверна СУБД, що вільно розповсюджується, Firebird Embedded Server [3].

Застосування модуля 1W-Rx6 дозволяє суттєво знизити витрати на модернізацію системи контролю температурного режиму старих зерносховищ. Він забезпечує підключення вже наявних та змонтованих аналогових датчиків температури до сучасної системи моніторингу, яка базується на цифрових технологіях.

Література

1. DS18B20 - Programmable Resolution 1-Wire Digital Thermometer [Електронний ресурс] – Режим доступу: https://www.analog.com/media/en/technical-documentation/data-sheets/DS18B20.pdf
2. ATtiny24/44/84 [Електронний ресурс] – Режим доступу: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7701_Automotive-Microcontrollers-ATtiny24-44-84_Datasheet.pdf
3. Using Firebird: (work in progress) [Електронний ресурс] – Режим доступу: https://www.firebirdsql.org/file/documentation/html/en/firebirddocs/ufb/using-firebird.html#ufb-cs-embedded

Рисунок В.2 – Тези доповідей на XXII конференцію, аркуш 2

ДОДАТОК Г

Запит на виконання кваліфікаційної роботи

SmartTerm™

ТОВ ЕЛЕМАН

Україна, м.Одеса, 65104, проспект Небесної Сотні, 101/10

Завідувачу кафедри
МЗКС, ФМФІТ
ОНУ імені І.І. Мечникова
д.т.н., професору
Малахову Є.В.

Шановний Євгеній Валерійович,

ТОВ «ЕЛЕМАН» звертається з проханням призначити здобувачу ступеня вищої освіти «бакалавр» за спеціальністю 123 «Комп'ютерна інженерія» ОНУ імені І.І. Мечникова Крушу Артуру Ігоровичу тему кваліфікаційної роботи «Апаратно-програмний комплекс моніторингу температурного режиму зерносховищ».

Означена тема відповідає актуальним потребам нашої компанії в контексті автоматизації процесів контролю за температурними умовами зберігання зернової продукції на промислових зерносховищах. Результати роботи будуть впроваджені в комплексі програмно-апаратних засобів SmartTerm2000 компанії.

Директор ТОВ ЕЛЕМАН Левенець С.В.

11.09.2024 р.

Тел. (067)3956823 (099)2037175
Mail to: smartterm@yahoo.com
info@smartterm.com.ua
WEB: www.smartterm.com.ua

ДОДАТОК Д

Довідка про впровадження

Olga Roznovets

Digitally signed by Olga Roznovets
Date: 2025.06.12 18:45:33 +03'00'

SmartTerm™

ТОВ ЕЛЕМАН

Україна, м.Одеса, 65104, проспект Небесної Сотні, 101/10

ДОВІДКА

про впровадження

Програмне забезпечення для модуля 1W-Rx6 на базі мікроконтролера ATtiny44, який адаптує термопідвіски з аналоговими датчиками до наявного комплексу моніторингу температури, розроблена студентом Одеського національного університету імені І.І. Мечникова Крушем Артуром Ігоровичем під час виконання кваліфікаційної роботи бакалавра на кафедрі МЗКС, успішно пройшла виробничі випробування в ТОВ «ЕЛЕМАН» та запланована до впровадження як складова моніторингової системи SmartTerm2000.

Директор ТОВ ЕЛЕМАН Левенець С.В.

03.06.2025 р.

Тел. (067)3956823 (099)2037175
Mail to: smartterm@yahoo.com
info@smartterm.com.ua
WEB: www.smartterm.com.ua