Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки, Факультет радіоелектроніки та телекомунікацій
(повне найменування інституту, факультету)

Кафедра мікро- та наноелектроніки
(повне найменування кафедри)

Пояснювальна записка
до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему Методика розробки HMI віртуального виробництва на базі TraceMode

Виконав: студент IV курсу, групи РТ-317

Спеціальності 153 Мікро- та наносистемна техніка
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Мікро- та наноелектронні прилади і пристрої

Іваницький І.С.
(прізвище та ініціали)
Керівник Василенко О.В.
(прізвище та ініціали)
Рецензент Загаєвська В.І.
(прізвище та ініціали)

2021

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет_ ІІРЕ, ФРЕТ
Кафедра____Кафедра мікро- та наноелектроніки
Ступінь вищої освіти _бакалавр
Спеціальність __153 Мікро- та наносистемна техніка
(код і найменування)
Освітня програма (спеціалізація)_ Мікро- та наноелектронні прилади і пристрої
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
_д.т.н., доц. Сніжной Г.В._____
«_____» _____20__року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

_____Іваницькому Івану Сергійовичу_____
(прізвище, ім'я, по батькові)
1. Тема проєкту (роботи) Методика розробки HMI віртуального виробництва на базі Trace Mode
_____
керівник проєкту (роботи)___к.т.н., доцент Василенко О.В._____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «_17__»__травня___2021__ року №___
2. Строк подання студентом проєкту (роботи)___25 травня 2021_____
3. Вихідні дані до проєкту (роботи)_ одноконтурна система керування температурою, межі керування: від 15°С до 30 °С; SCADA-система Trace Mode v. 6.0; відстань зв'язку до 1 км.
_____
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати види забезпечення віртуального виробництва (VM), визначити місце HMI та SCADA на етапі автоматизованого виробництва (Computer-aided Manufacturing), виконати ТЕОПР, розробити методику проєктування HMI на базі Trace Mode, розробити доступний зразок.
_____
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____
_____
_____
_____

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	Василенко. О. В. доцент кафедри МіНЕ		
Нормоконтроль	Коротун А. В. доцент кафедри МіНЕ		

7. Дата видачі завдання «_____» _____20___року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка задачі дослідження	25.04-26.04	*Виконав*
2	Літературний огляд	26.04-29.04	*Виконав*
3	Розробка структурної схеми	29.04-05.05	*Виконав*
4	Вибір апаратного та програмного забезпечення	05.05-06.05	*Виконав*
5	Параметризація системи	06.05-08.05	*Виконав*
6	Розробка дослідного зразку та експерименту	08.05-20.05	*Виконав*
7	Оформлення пояснювальної записки	20.05-22.05	*Виконав*
8	Підготовка презентації	22.05-24.05	*Виконав*

Студент _____ ____Іванцький І. С._____
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ ___Василенко О. В._____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

ПЗ: 47 с., 20 рис., 2 додатки, 17 джерел.

Мета роботи - спроєктувати автоматизовану систему із HMI на базі міні - SCADA.

Методи дослідження - лінгвістичні, технічні та інформаційні .

Методи проєктування - програмні.

Дипломна робота складається з трьох розділів.

В першому розділі було проаналізовано види забезпечення віртуального виробництва, визначено місце HMI та SCADA на етапі автоматизованого виробництва.

У другому розділі була виконано техніко-економічне обгрунтування технічного проєкту.

У третьому розділі було розроблено методику проєктування HMI на базі Trace Mode v. 6.0.

АВТОМАТИЗАЦІЯ ТЕХНОЛОГІЧНОГО ПРОЦЕСУ, SCADA СИСТЕМИ, ЛЮДИНО-МАШИННИЙ ІНТЕРФЕЙС, МІКРОКОНТРОЛЕР, ТЕРМОРЕГУЛЯТОР.

ЗМІСТ

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ

CAD – комп’ютерна підтримка проєктування (англ. computer-aided design)

CAE – комп’ютерна підтримка виготовлення (англ. computer-aided manufacturing)

CAM – комп’ютерна підтримка інженерних розрахунків (англ. computer-aided engineering)

COM – модель компонентного об’єкту (англ. Component Object Model)

DCOM – розподілена COM (англ. Distributed COM)

ERP-система – система плаування ресурсів підприємства

HMI - людино-машинний інтерфейс (англ. Human-Machine Interface)

OLE – технологія зв’язку та впровадження об’єктів (англ. Object Linking and Embedding)

OPC – єдиний інтерфейс для управління об’єктами оптимізації і ТП (англ. OLE for Process Control)

PDM – система управління даними про виріб (англ. Product Data Management)

PLC – програмований логічний контролер (англ. Programmable Logic Controller)

RTU(ПЗО) – пристрій зв’язку з об’єктом (англ. Remote Terminal Unit)

SCADA – диспетчерське керування та збір даних (англ. Supervisory Control And Data Acquisition)

VM – віртуальне виробництво (англ. Virtual Manufacture)

АКРК – автоматизований комплекс розподіленого керування

АРМ – автоматизоване робоче місце

АСК – автоматична система керування

ЕОМ – електронна обчислювальна машина

КІС – корпоративна інформаційна система

ОС – операційна система

ТП – технічний процес

ВСТУП

Віртуальне виробництво є найбільш розвиненим способом автоматизації всіх процесів реалізації готового продукту: від його концептуального моделювання, до перевірки його техніко-економічних та ергономічних показників на стадії передпродажної підготовки, тобто автоматизувати всі етапи життєвого циклу продукту (“life cycle management”). Це дозволяє швидко задовольнити вимоги потенційних покупців та слідувати трендам розвитку матеріально-технічного середовища.

Однією з задач даного проєкту було адаптувати можливості існуючої системи збору та моніторингу під задачі людино-машинного інтерфейсу в надбудові PDM. Задачами PDM є узгодження даних, отриманих від спеціалізованого програмного забезпеченя на різних етапах життєвого циклу продукту та візуалізація результатів моделювання і проєктування технологічних процесів для прийняття швидкого та оптимального рішення людиною.

HMI в структурі VM має бути багатофункціональним, оскільки в VM входить етап CAM, в якому як правило багато параметрів для моніторингу та контролю в багатоконтурних системах. Тому необхідно підібрати SCADA із розвиненою бібліотекою та можливостями програмування.

Серед всіх видів забезпечення автоматизованого виробництва важливе місце посідає методичне, яке гарантує повторюваність дій та надійність результату при розробці аналогічних систем.

Для перевірки адекватності розробленої методики необхідно розробити дослідний зразок, наприклад, одноконтурної системи автоматичного керування, при чому із мінімальними витратами на апаратне забезпечення.

1 ВИДИ ЗАБЕЗПЕЧЕННЯ ВІРТУАЛЬНОГО ВИРОБНИЦТВА

1.1 Опис АСК ТП

Перебіг будь-якого технологічного процесу – є певна алгоритмічно задана зміна параметрів процесу в часі та / або в просторі, тобто зміна стану технологічної системи. Отже, будь-який технологічний процес повинен супроводжуватись інформацією про послідовність змін стану процесу в часі та / або просторі [1].

Інформаційні потоки зароджуються на рівні керування обладнанням (рис. 1.1): показники випуску продукції, витрат сировини, енергії, води та інше, вимірюються датчиками і обробляються контролерами.

Керування виробничим процесом виконують АСК ТП, нижній рівень яких займається безпосереднім керуванням технологічними процесами та обладнанням, а на верхньому рівні є система диспетчерського керування одним із складників є SCADA [2].

Наступним рівнем є рівень керування виробництвом і планування ресурсів підприємства.

Точна, своєчасна та достовірна інформація на кожному рівні виробництва дозволяє оцінити витрати, якість і конкурентоспроможність-ність продукції, ефективно організувати керування підприємством в сучасних умовах господарювання.

Традиційний підхід виділяє в системах промислової автоматизації чотири рівні:

1. Ввід / вивід:

• вимірювальні пристрої – датчики;

• керуючі пристрої – виконавчі механізми.

2. Керування вводу / виводу – контролери:

• прості пристрої збору та обробки інформації;

• програмовані логічні контролери;

• промислові комп'ютери.

3. Системи керування технологічним процесом – це системи диспетчерського керування та збору даних, що представляють, в загальному випадку головним сервером і мережею автоматизованих робочих місць (АРМ) операторів.

4. Системи керування фінансової, господарської та адміністративної діяльності підприємства.

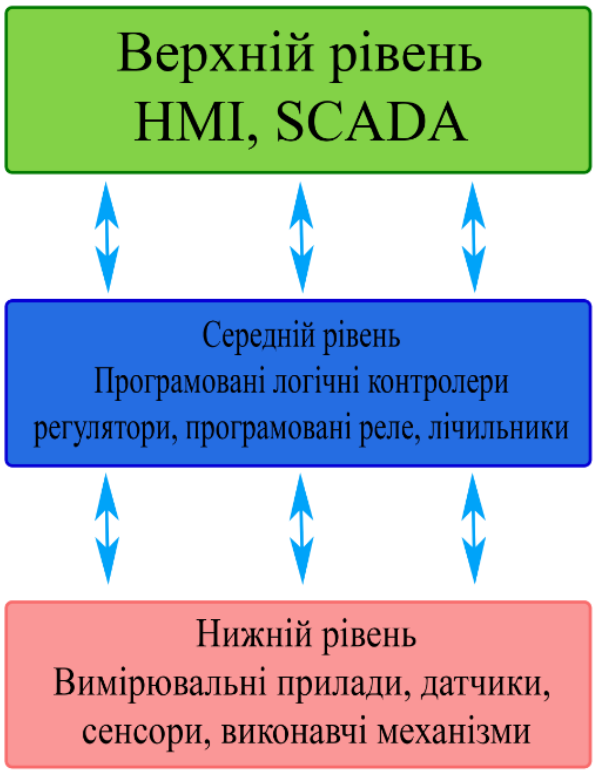

Рисунок 1.1 – Рівні керування обладнанням

Створення систем промислової автоматизації, в сучасних умовах, не може розглядатися у відриві від проблеми комплексної автоматизації підприємства. Система промислової автоматизації повинна бути складовою частиною інтегрованої корпоративної інформаційної системи (КІС). Тільки при такому підході можна забезпечити інформаційну «прозорість» підприємства та формувати єдину інформаційну мережу підприємства, що приймає стратегічні рішення. Вся інформаційна система працює для забезпечення інформаційної підтримки прийняття цих рішень (рис. 1.2) [1].

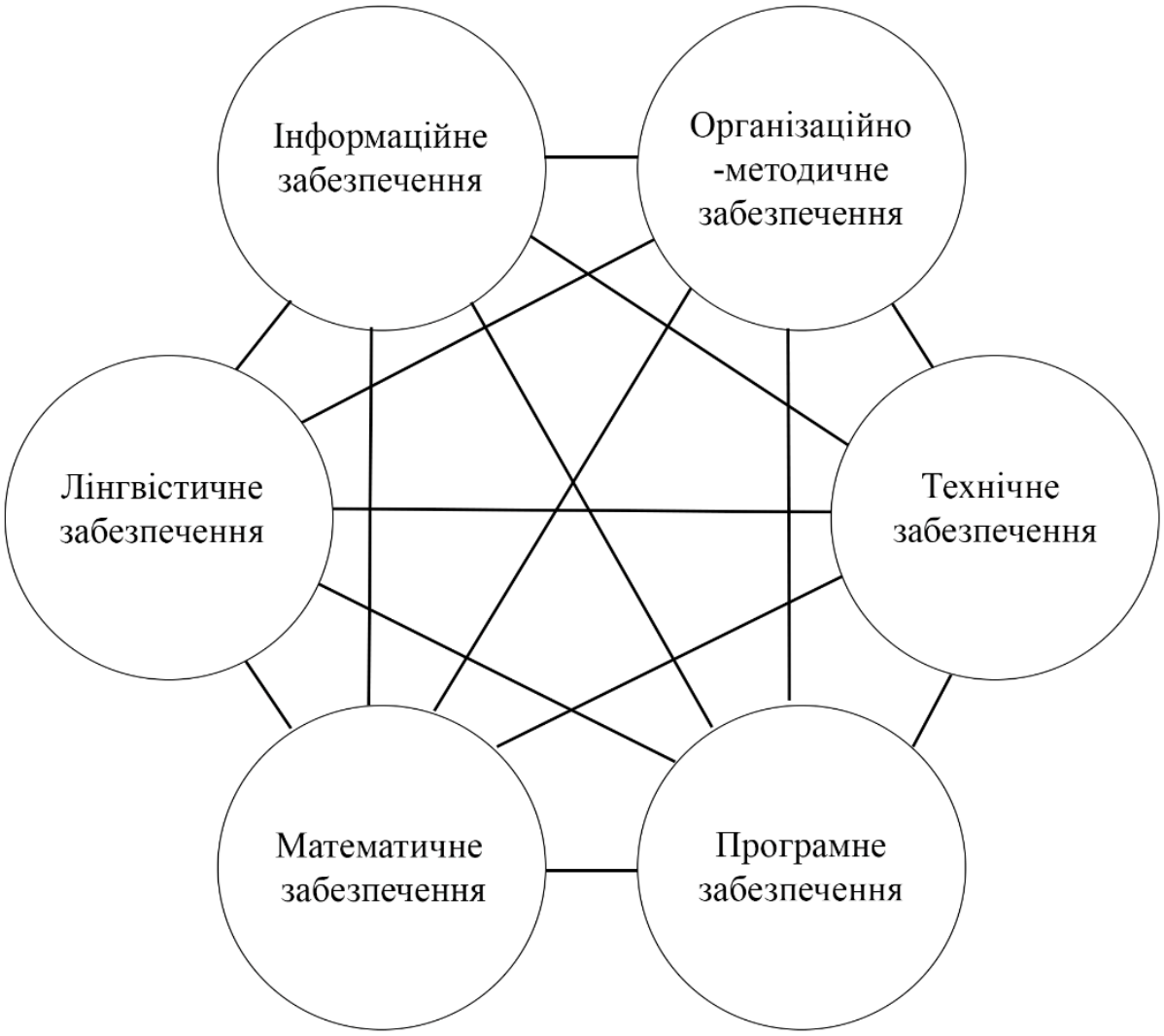

Рисунок 1.2 – Забезпечення інформаційних систем

Центральним елементом в будь промисловій системі служить обчислювальний блок, який, в залежності від розв'язуваної задачі, може бути або найпростішою мікроплатою, або багатопроцесорним комплексом із зовнішньою пам'яттю великого об'єму, базою даних і засобами мережевої взаємодії. Обчислювальний блок вирішує два завдання. Перше – це власне програмне керування на основі моделі реального процесу. Друге – організація інтерфейсу з обслуговуючим персоналом. Тут візуалізується стан об'єкта керування шляхом виведення його параметрів і статистичних даних, а також містяться засоби для ручного керування [3].

В умовах комплексної автоматизації, процес керування виконують програмовані логічні контролери, які обмінюються інформацією з

технологічним процесом через свої виходи і входи. Контролери, як правило, об'єднані в локальну мережу для того, щоб була можливість контролювати і керувати кожним з них з головною ЕОМ.

Однак, як відомо, ніяке виробництво не може обійтися без участі людини. Тільки людина здатна реально проаналізувати поточний стан виробничого процесу. Тільки людина може прийняти рішення при виникненні непередбаченої, неординарної ситуації. Повністю замінити людину комп'ютером неможливо. Тому на сучасних виробництвах створені автоматизовані робочі місця (АРМ) на основі персональних комп'ютерів для супервізорного керування виробництвом. Кожен оператор зі свого робочого місця має доступ до даних, що зберігаються в головній ЕОМ по мережі персональних комп'ютерів, і може як контролювати їх, так і змінювати, тобто управляти технологічним процесом [4].

На рис. 1.3 наведено приклад рівнів керування сучасного автоматизованого комплексу розподіленого керування (АКРК), або, як в подальшому будемо називати, автоматизованої системи керування технологічним процесом на виробництві.

У розвитку промислових систем автоматизації в основному розглядаються загальні тенденції комп'ютерної індустрії, однак можна вказати кілька принципових особливостей, які вимагають спеціалізованих рішень [5].

1. Промислові системи функціонують у важких для електронної техніки умовах зовнішнього середовища, тому в порівнянні із звичайними комп'ютерами, вони повинні мати підвищену термо-, вібро-та ударостійкість.

2. Потрібно підключати набагато більш широку номенклатуру зовнішніх пристроїв.

3. Час реакції системи на зміни параметрів об'єкту керування визначається зовнішніми реальними часовими інтервалами – такі системи називаються системами реального АСК. Для особливо відповідальних додатків, наприклад при керуванні літаком, реакція повинна бути практично

миттєвою. Це, зокрема, припускає підвищену надійність і апаратної, і програмної частин.

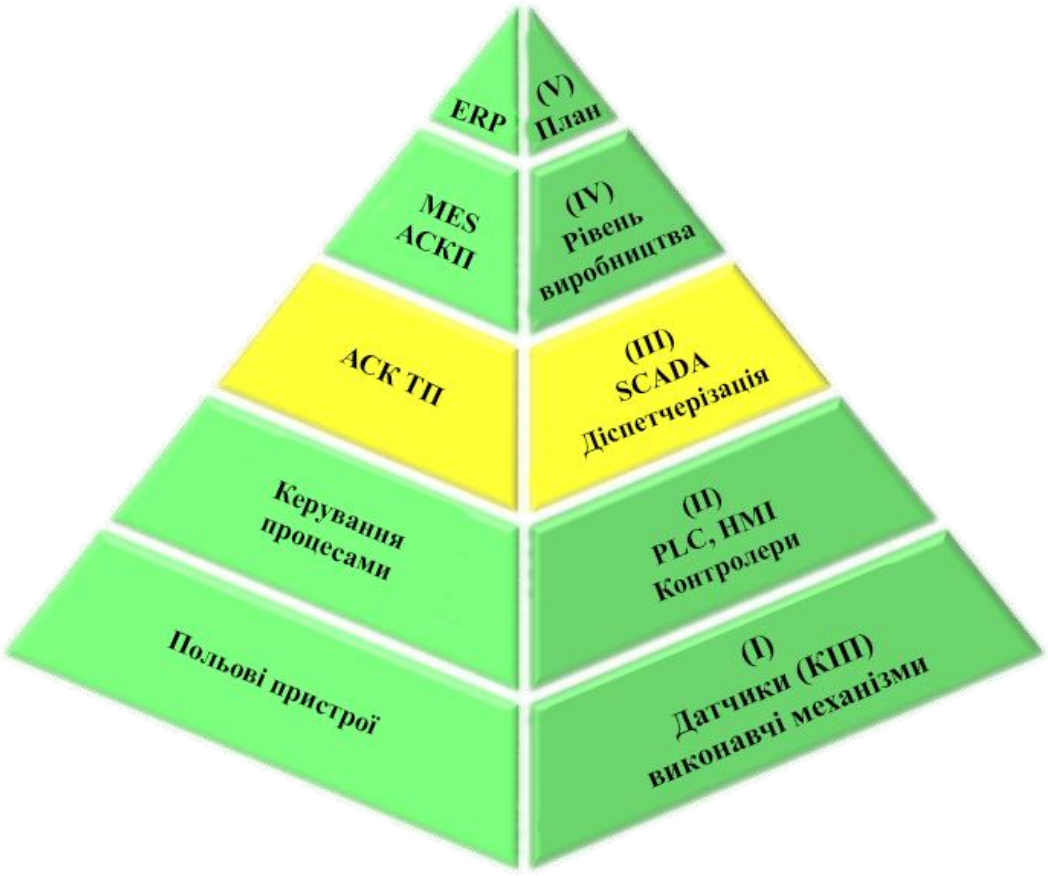

Рисунок 1.3 – Рівні керування АСК ТП

Створення автоматизованих робочих місць:

1. дозволяє замінити громіздкі, незручні, дорогі щити керування на персональні комп'ютери. Це зменшує робочу площу, робить робоче місце більш зручним, простим, наочним і значно знижує його вартість;

2. робить систему керування гнучкішою, коли для того, щоб додати кілька сигналізаційних ламп або керуючих важелів не потрібно повністю міняти весь щит, а можна лише зробити невеликі зміни в комп'ютерній програмі;

3. полегшує аналіз і керування технологічним процесом, адже програмне забезпечення дозволяє представляти дані в найрізноманітнішому

вигляді (графіки, діаграми, рівні, зміна положення, кольору і т.д.), дозволяє зберігати дані тривалий час, а керування здійснюється натисканням клавіші або кліком мишки.

Створення сучасних АСК — досить складний процес, що вимагає скоординованих дій різних груп фахівців, які ведуть прикладне проєктування систем керування. Подібна технологія дозволяє максимально задовільнити індивідуальні запити споживачів, багаторазово скоротити час налагодження комплексу безпосередньо на об'єкті і вчасно навчити персонал роботі з АСК. Можна виділити декілька стадій виконання основних етапів робіт, які регламентуються Держстандартами (рис. 1.4) [1].

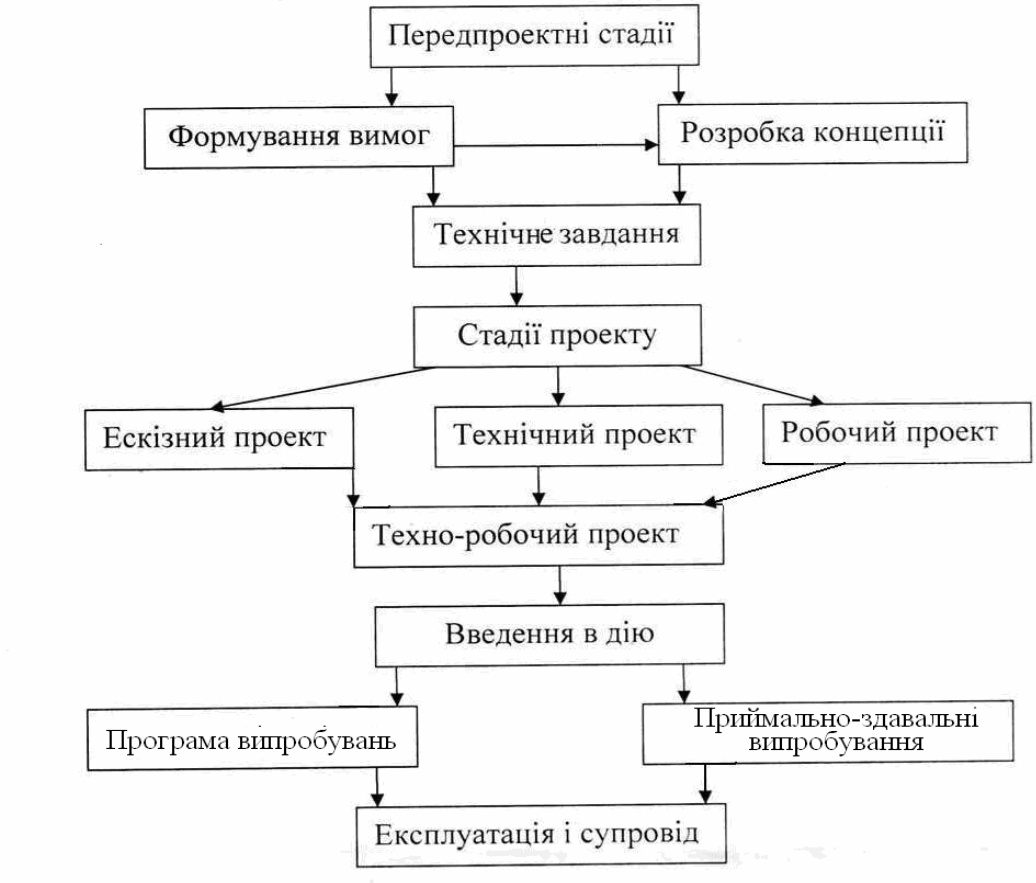

Рисунок 1.4 – Стадії проєктування АСК ТП

1.2 SCADA-системи

SCADA - це абревіатура від Supervisory Control And Data Acquisition - диспетчерське керування і збір даних. Системи SCADA складаються з мереж, електронних приладів, датчиків вимірювальних приладів, комутаційних пристроїв тощо. Структура системи дозволяє контролювати і керувати процесами локально або віддалено, аналізувати дані, генерувати сигнали тривоги та надсилати їх у різні місця тощо. Термін SCADA часто використовується для визначення систем, які локально розосереджені. Система складається з центрального хоста або ведучого, одного або декількох збірників даних, блоків керування, також відомих як RTU, і певного стандартного та / або нестандартного програмного забезпечення, що використовується для моніторингу та керування дистанційно розташованими елементами даних поля [7-8].

1.3 Вимоги до SCADA-системи

Автоматизація зменшила кількість локально прив’язаних працівників на підприємствах, а також знизила ризик людської помилки, але вона не може здійснювати керування та операції без правильної параметризації та конфігурації програм та пристроїв. Для того, щоб впоратися з цими речами обслуговування персонал диспетчерської все ще повинен добре сприймати систему. В результаті розвитку технологій з'явилися нові вимоги до моніторингу та контролю за показниками на усіх рівнях виробництва (рис. 1.5.) [9].

Зміни у системі слід відстежувати та контролювати дуже ретельно, а результати та події потрібні для автоматичного аналізу, потім це дозволяє швидко реагувати на системні помилки. Це створює певний тиск на програми та постачальників пристроїв. У наш час ситуація на ринку досить складна,

через велику кількість постачальників SCADA-систем різного плану та направленості. Це створює потребу в постійному розвитку систем контролю для постачальників та моніторингу ринку для майбутніх юзерів.

Якщо значення вимірювання є поганим, це може спричинити значні помилки протягом тривалого періоду АСК. Неправильне вимірювання даних спричиняє помилки у щоденних, щомісячних та річних звітах. Оператори також можуть мати труднощі з прийняттям рішення про відновлення роботи пристроїв, які, можливо, були автоматично відключені інтелектуальними електронними пристроями особливо якщо оператори не мають чіткої перевірки системних умов або пов'язаної топології .

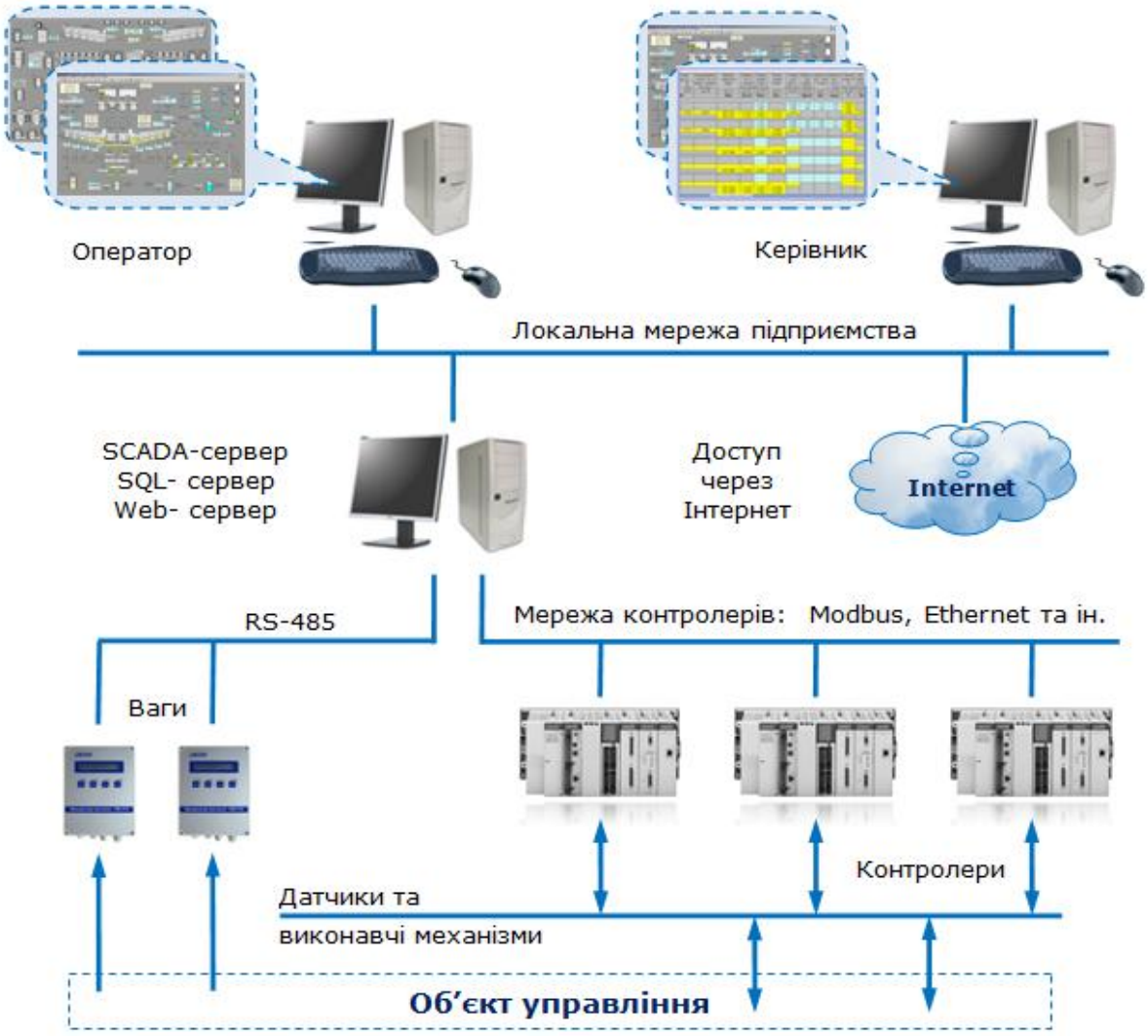

Рисунок 1.5 – Стурктура застосування SCADA-систем

Проблема SCADA полягає в тому, як раціонально використовувати дані. Старі ПЗО не призначені для відстеження аналогових вимірювань у реальному часі. Застосовується лише звітування за винятком, коли певні порогові значення перевищуються. Під час електричних несправностей, напр. коротке замикання, аналогові сигнали змінюються дуже швидко, і лише детальніше відстеження може дати інформацію про фазові залежності та динамічні зміни величин сигналу. Якщо використовуються нові ПЗО, це дає більше можливостей реагувати на несправності і отримати достовірну інформацію. Однією з основних проблем є також несумісність пристроїв та відсутність відповідного стандарту, який би уніфікував форми даних, що створюються різноманітними пристроями. Інформація від пристроїв відрізняється за протоколами і з цієї причини непросто об’єднати пристрої для зв’язку між собою [10-11].

1.4 Віртуальне виробництво

Віртуальне виробництво-це передова форма автоматизованого виробництва. У 1990 році з'явилися концепції віртуального світу і віртуальних середовищ. Віртуальна реальність визначається як комп'ютерне інтерактивне та імерсивне 3D-середовище, що імітує реальність. Термін "віртуальна машина" вперше з'явився в рамках ініціативи Міністерства оборони США з віртуального виробництва. Як концепція, так і термін останнім часом отримали широке міжнародне визнання і розширилися за своїм охопленням. У першій половині 1990-х років новаторська робота в цій галузі була проведена кількома великими організаціями, в основному в аерокосмічній, землерийній та автомобільній промисловості, а також кількома спеціалізованими академічними дослідницькими групами. Останнім часом став очевидний зростаючий інтерес світового ринку підвищенням обізнаності про величезний потенціал віртуального виробництва. У

технологіях і організації виробництва віртуальна реальність стала основою віртуального виробництва, спрямованого на задоволення очікувань користувачів/покупців продуктів, а також щодо їх низької вартості і часу виконання. Віртуальне виробництво включає в себе швидке вдосконалення виробничих процесів без використання фонду робочого часу машин [12].

Віртуальне виробництво можна розділити на три групи відповідно до типу продукту та дизайну процесу:

1. Віртуальне виробництво, орієнтоване на дизайн - на етапі проєктування проєктно-орієнтована віртуальна машина надає інженеру-проєктувальнику виробничу інформацію. Деталі або вся машина моделюються і оцінюються для перевірки технологічності і здатності до складання. Мета полягає в тому, щоб оптимізувати дизайн продукту і проєктування технологічних процесів з допомогою моделювання виробництва для досягнення якості та гнучкості.

2. Орієнтоване на виробництво віртуальне виробництво - технологія моделювання використовується у виробництві планування або нова модель процесу для оцінки та оптимізації виробничих процесів на основі інтегрованого процесу виробництва продукту Розробка і перевірка раціональності і продуктивності технологічного процесу.

3. Віртуальне виробництво, орієнтоване на управління - технологія аналізу застосовується до моделі управління для моделювання діяльності з управління виробництвом виробничої лінії або цеху, щоб реалізувати оптимальне управління на основі моделей і фактичного процесу.

Однією із задач роботи було визначити місце SCADA у структурі віртуального виробництва, нами запропоновано структуру на рис. 1.6, де SCADA допомагає забезпечити етап CAM і пов’язане з PDM.

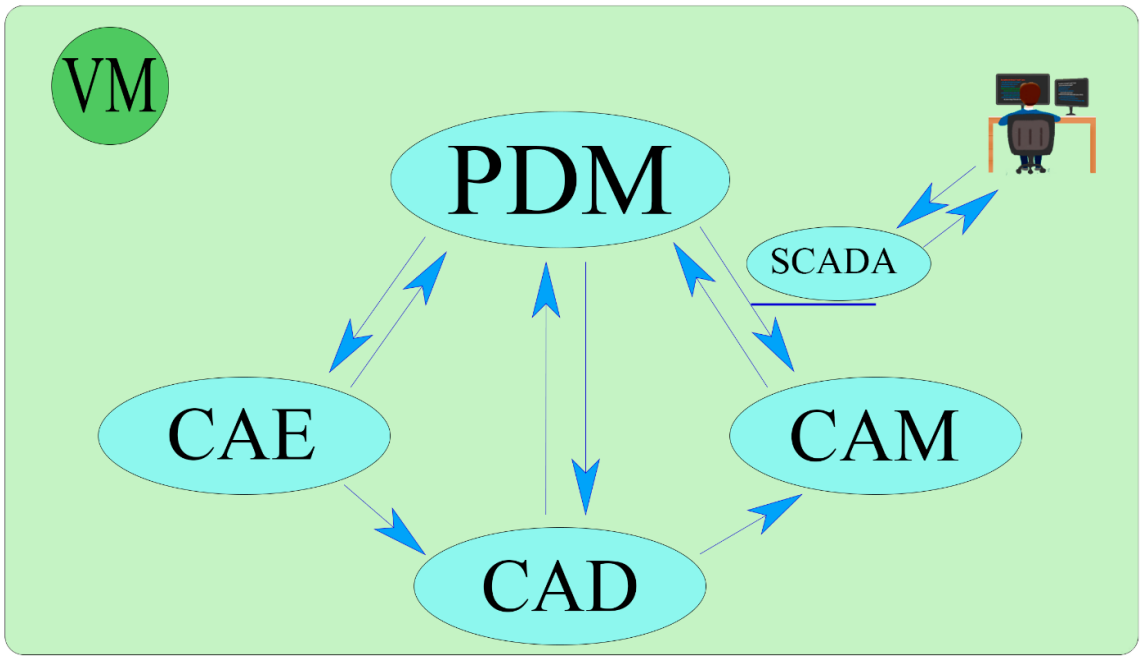

Рис. 1.6 – Місце SCADA у віртуальному виробництві

1.5 Програмовані логічні контролери

Програмований автомат - це пристрій, в якому залежність виходу від входу задається програмою. Тому ми можемо використовувати один ігровий автомат для тисяч різних додатків. Це система керування, розроблена спеціально для промислового застосування. Найчастіше його називають абревіатурою PLC (програмований логічний пристрій). Ми можемо розділити їх за кількома аспектами. Кожна група має свої переваги і недоліки, і тому для конкретного додатка необхідно добре розглянути, до якого типу ПЛК ми прагнемо.До основних переваг ПЛК можна віднести можливість швидкого впровадження системи, надійність, діагностику і нескінченні зміни призначення. Для застосування необхідно спроєктувати відповідні компоненти, їх підключення, розробку і налагодження програми керування. Реалізована складна самодіагностика гарантує стабільність і безпеку системи. Можливість зміни програми в процесі використання є величезною перевагою. На практиці вони рідко твердо дотримуються початкової пропозиції щодо вирішення проблеми. Навпаки, часто під час налагодження

виявляються нові факти, які не були передбачені у вихідному рішенні, тому необхідно керувати програмою для налаштування або завдяки використанню нової технології для гнучкого реагування [13].

Однак кожна додаткова зміна тягне за собою ускладнення і ризики, пов'язані зі складністю програми та її загальною виразністю. Для систем з фіксованою логікою, таких як релейні системи, додаткова зміна, часто є великою, а іноді практично нерозв'язною проблемою.

1.5.1 Класифікація контролерів.

Програмовані автомати випускаються в різних варіаціях. За розміром їх можна розділити на дві основні групи.

1. Компактний ПЛК.
2. Модульний ПЛК.

Представники цієї групи є найменшими і дешевими програмованими автоматами. Вони мають фіксоване число входів і виходів. Часто вони включають в себе РК-дисплей, за допомогою якого ви можете стежити за станом програми і навіть програмувати все це. Вони зазвичай замінюють більш ранню логіку реле різних машиних механізмів. До базового модулю можна підключити один або кілька модулів розширення з фіксованою кількістю і типом входів і виходів. Їх найбільшою перевагою, в порівнянні з модульними, є більш низька ціна. У цю групу входять, наприклад, Simatic S7 - 200 від Siemens.

Модульні або модульні ПЛК збираються з окремих модулів. Так що це величезна свобода і можливість індивідуально налаштувати необхідну конфігурацію входів для конкретного додатка, виходів, комунікаційних модулів

1.6 Людино-машинний інтерфейс

Це графічний інтерфейс оператора між людиною і виробничим процесом, що дозволяє контролювати даний процес. Кожен пристрій HMI повинен мати реалізовану в ньому ОС і, як правило, драйвер зв'язку для використовуваного ПЛК.

На практиці вони зазвичай реалізуються операторськими панелями, розташованими безпосередньо на заводі (автономними або інтегрованими в електричний розподільний щит). Системи HMI не обмежуються встановленими на заводі операторськими панелями, але кожна SCADA або ERP - система може бути позначена як інтерфейс HMI [3].

1.7 OPC сервер

З ростом автоматизації процесів керування збільшується і обсяг даних, які необхідно передавати між системами. Клієнтські станції, які збирають ці дані, використовують їх, наприклад, для візуалізації процесів керування. Існує більше можливостей для забезпечення надійної передачі даних між цими системами. Звичайним способом було використання точно побудованих драйверів, застосовних до конкретних пристроїв, що принесло з собою кілька недоліків:

1. кожна програма повинна включати драйвер для конкретного обладнання, яке він використовує;

2. якщо існує невідповідність між драйверами від різних постачальників;

3. зміна властивостей обладнання призводить до збою драйвера ;

4. іноді виникають конфлікти при доступі до обладнання: два різних пакети програмного забезпечення не можуть спільно використовувати пристрій, якщо кожен з них не містить незалежний драйвер;

Прагнення усунути ці недоліки призвело до створення терміна OPC. Їх обслуговуванням займається організація OPC Foundation, що об'єднує багатьох виробників продукції для контрольно-вимірювальної апаратури. Для цієї бакалаврської роботи найцікавішою специфікацією є доступ до даних OPC, який використовує міжпроцесний зв'язок, розроблену Microsoft, а саме модель OLE/COM/DCOM (Зв'язування та вбудовування об'єктів/компонентна об'єктна модель / розподілений com). В принципі, доступ до даних OPC описує зв'язок з двох сторін - сервера і клієнта. Сервер піклується про співпрацю з конкретними даними, з цього пристрою передаються клієнтам, які в подальшому обробляють ці дані (рис. 1.7) [12].

Запис OPC не повинен реалізовувати будь-які запропоновані інтерфейси, однак вона повинна мати в наявності наступні параметри:

1. поточне значення (Current Value);

2. позначка АСК (Time Stamp) - час останнього оновлення елемента;

3. якість поточного значення (Quality) – достовірність прав доступу до поточного елементу;

4. права доступу (AccessRights) - вказує параметри читання / запису для власного типу даних;

5. власний тип даних (Native Datatype) - тип інформації в структурі даних;

6. максимальна частота оновлення сервера (Server Scan Rate);

7. унікальний номер на сервері OPC (Server Handle);

8. введення унікального номеру в OPC клієнта (Client Handle);

9. Признак активності(Activity flag) - вказує на активність / бездіяльність елементу;

10. Необхідні типи данних -визначає тип інформації в структурі даних, яку клієнт запитує для поточного значення.;

11. Додатковий ідентифікатор тега (Blob) - дозволяє швидше вводити і скасовувати елементиПерші шість атрибутів передаються клієнту при виклику методів iopcitemproperties.

Iopcgroupstatemgt-служить клієнту для додаткової установки атрибутів групи OPC. Група ініціалізується, коли вона створюється методом addgroup з розділу iopcserver.

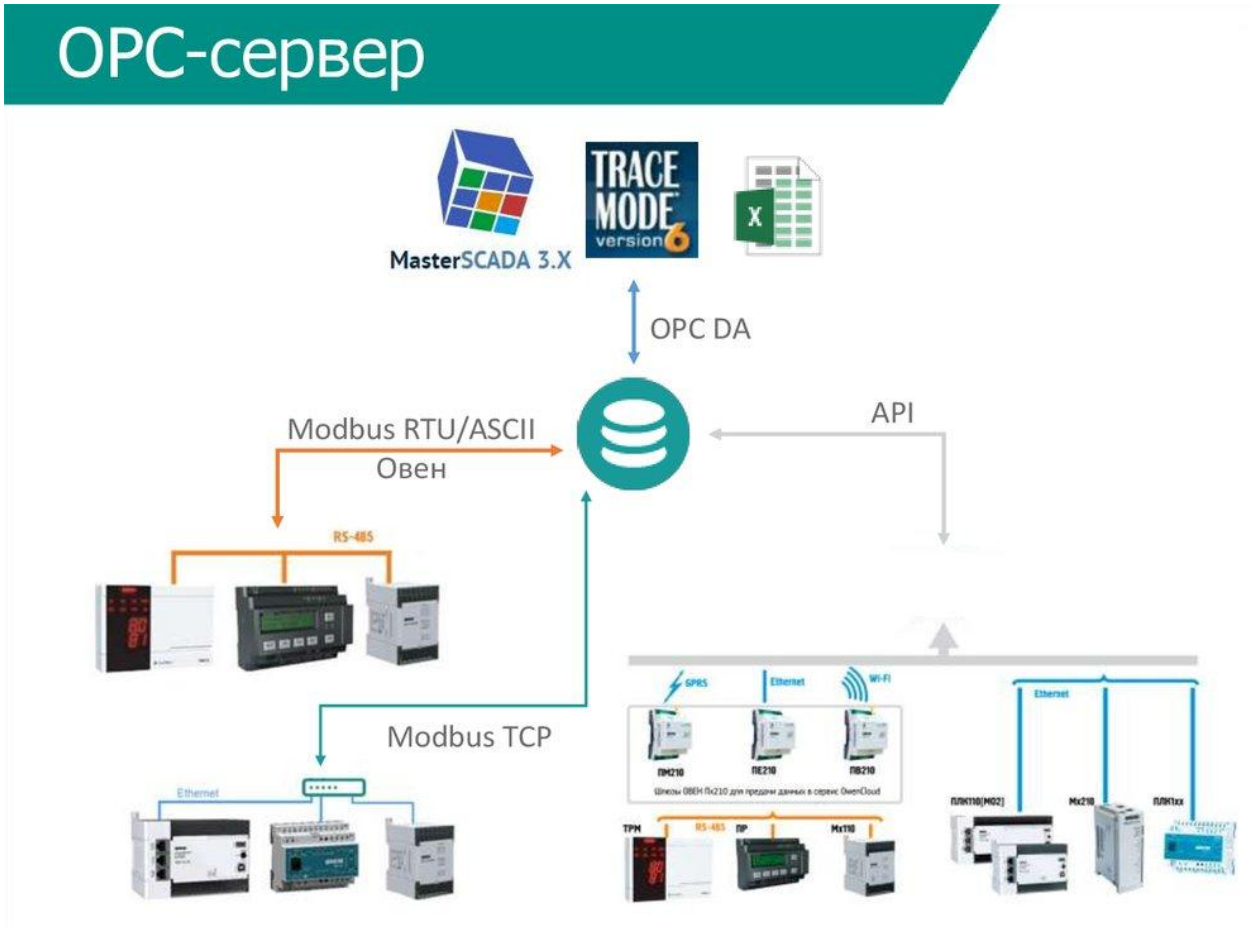

Рисунок 1.7. – Використання OPC серверу

Основними атрибутами групи OPC є ім'я групи (name), частота оновлення (Update rate), мертва смуга (Deadband), часова база (TimeBase), локалізація (lcid), Активний прапор (active flag), дескриптор сервера (servergrouphandle) і дескриптор клієнта (clientgrouphandle). Цей інтерфейс визначає кілька методів, за допомогою яких ви можете працювати з атрибутами [13].

Iopcitemmgt - цей інтерфейс являє собою основний інструмент для роботи з елементами. Елемент однозначно ідентифікується не по імені, а за номером, який генерується як сервером, так і клієнтом при створенні

елемента. Iopcsyncio-читання / запис даних в Елементи. Як читання, так і запис виконуються синхронно, тобто потік методу цього інтерфейсу очікує завершення операції сервером [13].

Iopcsyncio2 - на відміну від iopcsyncio, методи цього інтерфейсу виконуються синхронно. Потік, який запускає операцію, не чекає її завершення. Коли операція завершена, сервер використовує вихідні методи інтерфейсу iopcdatacallback для інформування клієнта про результати операції. Цей інтерфейс реалізує методи read, write, refresh2, cancel2, setenable, getenable [13].

1.8 Технологія COM

Модель компонента COM (communications port) є одним з варіантів міжпроцесної взаємодії в Windows. Ця технологія вперше з'явилася в 1995 році і, таким чином, стала однією з основ для подальшої розробки програмних додатків в MS Windows. Технологія COM також стала основою для багатьох інших, таких як:

1. DCOM що надають послуги компонентного зв'язку на віддалених комп'ютерах;

2. COM+, побудований на COM і DCOM з використанням додаткових сервісів, таких як MTS (Microsoft Transaction Server), середовище, що вирішує проблеми одночасних операторів;

3. OPC, галузевий стандарт, побудований на COM, який використовується для посередництва та архівування даних в управлінні технологічними процессами;

Під поняттям компонента можна представити чітко відокремлену частину програми. Кожна програма компонента (COM-сервер) надає послуги одному або декільком додаткам (COM-клієнтам), використовуючи екземпляри своїх компонентів. Щоб і сервер, і клієнт могли знайти спільну

мову, вони повинні говорити на правильній мові, який описує модель COM. Кожен компонент COM можна логічно розділити на дві основні частини: інтерфейс і реалізацію. З точки зору мови програмування, інтерфейс являє собою список методів і визначень реалізації потужних органів цих методів. Потім клієнтський додаток отримує тільки покажчик на інтерфейс даного екземпляра компонента всередині сервера, нічого не знаючи про реалізацію цього інтерфейсу. Кожен абстрактний клас має таблицю віртуальних функцій, і для кожного АСК є тільки одна, яка є спільною для всіх екземплярів цього АСК. Потім кожному екземпляру належить покажчик на таблицю. Потім таблиця містить покажчики на реалізації всіх віртуальних методів АСК. У фіналі клієнт отримує покажчик на покажчик на таблицю віртуальних функцій. Оскільки можна створювати компоненти і клієнти на різних мовах програмування, необхідно створити так званий двійковий сумісний інтерфейс. Для того, щоб оголошення C++ відповідало правилам COM, його все одно необхідно змінити:

1. угоди про виклик __stdcall-повідомляє, як параметри будуть передані функції;

2. повернення значення функції типу HRESULT;

3. реалізація інтерфейсу IUnknown ;

Повертається значення HRESULT-це 32-розрядне число, що містить код, що описує успіх методу. Кожен COM-інтерфейс є похідним від базового інтерфейсу, так званого інтерфейсу IUnknown, який містить рівно три методи: Addref (), release () і queryinterface. COM визначає правила, які керують часом життя об'єкта. Перше правило полягає в тому, що якщо ми створюємо ще одне посилання на об'єкт, повинен бути викликаний метод Addref (). У другому говориться, що якщо, навпаки, посилання на об'єкт скасовано, необхідно викликати метод release (). Метод queryinterface використовується для динамічного перегляду інтерфейсу. Після вказівки імені інтерфейсу покажчик повертається до цього інтерфейсу. Якщо він не існує, він повертає код помилки.

Кожен компонент потребує “квартири для свого життя”, яка може бути двох типів-STA (однопотокова квартира), яка може містити тільки одне потужне ядро, і MTA (багатопотокова квартира), яка дозволяє декільком потокам використовувати методи компонентів одночасно. Важливою особливістю є так зване сортування параметрів. Це перетворення переданих аргументів, в правильну форму (перетворення little/big endian, кодування чисел і т.д.). З'єднання COM-клієнта і сервера забезпечується двома проміжними ланками-проксі і заглушкою.

2 ВИБІР АПАРАТНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Обґрунтування вибору апаратного забезпечення

2.1.1 Вибір контролеру

На сьогоднішній день індустрія радіо- приладобудування пропонує великий спектр різноманітного обладнання для будь якого споживача. Це надає нам змогу вибрати те обладнання яке більш зручне для використання саме для нашого проєкту.

Оскільки розроблена система є дуже простою(контролюється лише один параметр), для нього не потрібні PLC які хоч є більш потужними і легшими для управління, але економічно не доцільні у порівнянні з масовими мікроконтролерами по типу Arduino Uno який і буде використовуватись у даному проєкті (рис. 2.1).

Arduino Uno це відкрита електронна платформа, заснована на простому апаратному і програмному забезпеченні користувача. Фізично ця платформа складається з друкованої плати з мікроконтролером на базі ATmega328. Він включає в себе 14 цифрових входів і виходів (6 з них можуть використовуватися в якості ШІМ-виходів), 6 аналогових входів і кварцовий генератор 16 МГц, USB-з'єднання, а також кнопку сбросу. Мікроконтролер на платі Arduino програмується з використанням спеціальної мови програмування у власному середовищі розробки Arduino, або з використанням спеціального софту для програмування мікроконтролерів.

Рисунок 2.1 - Arduino Uno [14]

2.1.2 Вибір датчика температури

Створення проєкту потребувало варіант датчику, перевагами якого є доступність та низька ціна та який працює в діапазоні температур 15-50°C з невеликою точністю +-0.5°C. Був обраний датчик DS18B20, оскільки він задовольняє усім поставленим вимогам.

Датчик температури DS18B20 - це повністю інтегрований продукт, який дозволяє проводити незалежне від часу вимірювання температури в діапазоні від -55° C до + 125° C . Датчик дозволяє вимірювати температуру з роздільною здатністю до 2-4° C, що становить 0,0625 ° C [15]. Однак для виконання цієї роботи достатньо дозволу 2-2, яке становить 0,25 ° C. кожному виробленому чіпу DS18B20 присвоюється унікальний 64-бітний серійний номер. Цей номер дозволяє ідентифікувати окремі датчики при послідовному підключенні декількох датчиків [16]. DS18B20 підтримує два режими живлення:

1) Паразитний (рис. 2.2): живлення датчика здійснюється тільки за допомогою сигналу (1-провідна шина).

2) Зовнішнє джерело живлення (рис. 2.3): датчик живиться від додаткового джерела напруги.

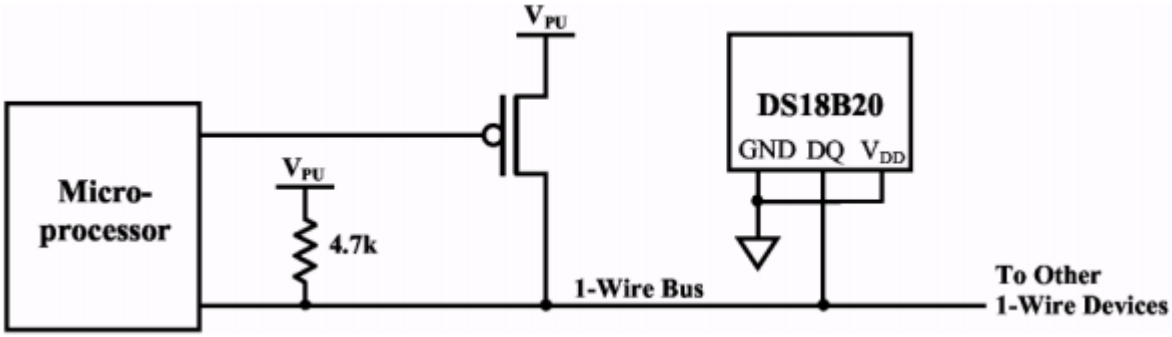

Рисунок 2.2 - Паразитне живлення DS18B20 під час перетворення температури

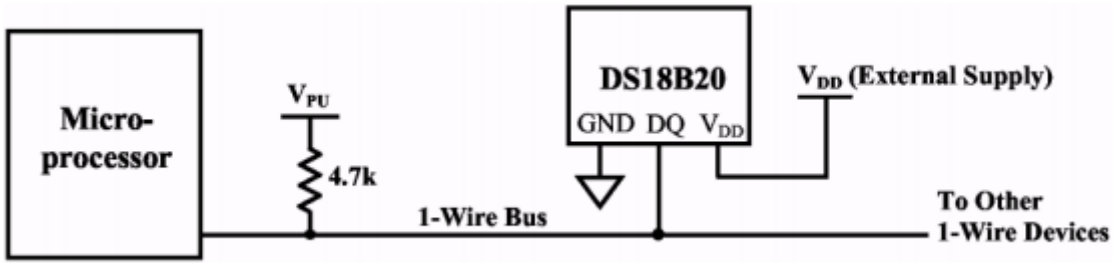

Рисунок 2.3 – Живлення DS18B20 від зовнішнього джерела

2.2 Обґрунтування вибору програмного забезпечення

Для програмування контролера було обрано FLProg, оскільки ця програма значно полегшує роботу з Arduino та доступна у мережі у вільному доступі. Розробник цієї програми працював з АСК ТП і вирішив створити софт для роботи з різного типу мікроконтролерами зібравши найбільш цікаві та корисні функції з інших інтегрованих середовищ програмування, таких як Tia-Portal, Zelio Soft, Logo Soft Comfort і т.п. .

Програма дозволяє збирати схеми у двох типах:

1. Функціональні схеми (FBD) – мова функціональної блок-схеми є однією з графічних мов. Він виражає поведінку функцій, функціональних блоків та програм як набір взаємопов'язаних графічних блоків, подібних до електронних схемних схем. Це система елементів, які обробляють сигнали. Тут часто використовуються стандартні функціональні блоки, такі як бістабільні елементи (пам’яті з домінуючим увімкненням чи вимкненням, семафори), елементи для виявлення піднімаються та падаючих фронтів, лічильники, таймери та комунікаційні блоки, визначені стандартом IEC 11315. Як і у випадку з LAD, легко зрозуміти в Інтернеті, чому умова виконана чи ні [10].

2. Релейні схеми (LAD) – програма складається із східчастих діаграм, які іноді також називають релейними діаграмами. Тут використовуються релейні контакти. На перший погляд це дуже чітко. Однак складні побудови

програм тут уже не можна створювати так легко, і програма стає заплутаною. В онлайновому режимі легко зрозуміти, чому певна умова виконується чи ні [10].

Такий спосіб програмування виявився дуже зручним для легкого входження в розробку систем АСК інженерів-електриків і електронників. Розробляючи проєкти пристроїв, вони можуть легко прив'язати роботу цих установок до алгоритмів роботи контролера.

При виборі програмного пакету SCADA треба приділяти увагу таким параметрам, як зручність у використанні, мовна локалізація, бібліотека модулів для виведення на екран наглядного процесу виробництва, зручний інтерфейс, підтримка різних типів підключення приладів, кількість методичних матеріалів з роботи програми, якість праці технічної підтримки та цінова політика виробника.

За цими параметрами було обрано пакет Trace Mode IDE 6, він є інтуїтивно зрозумілим у використанні, має велику бібліотеку файлів, може похизуватись великою кількістю посібних матеріалів з використання, починаючи з довідки у самій програмі, закінчуючи великою кількістю відео на YouTube-каналі розробників, який постійно поповнюється відео-інструкціями починаючи з 2009 року.

Автономні системи в TRACE MODE володіють розвиненим функціоналом. Вони здатні працювати в умовах поганої зв'язку (телемеханіка) і виконувати наступні операції:

1. збір даних 24x7 з більш ніж 2678 ПЛК, лічильників та пристроїв через безкоштовні драйвери або OPC;

2. візуалізація інформації реального часу на якісному графічному операторському інтерфейс (SCADA/ HMI);

3. операторське керування процесом;

4. виконання керуючих програм, написаних на 5-ма мовами стандарту МЕК 6-1131/3, по розкладами, зі статистичними даними, або за допомогою рецептів;

5. моніторинг і управління тривожними і попереджувальними повідомленнями (алармами). Розсилка попереджень електронною поштою та SMS;

6. ведення історії процесу у власній промислової СУБД;

7. створення якісних звітів, за допомогою власного генератора;

8. забезпечення безпеки SCADA, що відповідає сучасним вимогам;

3 РОЗРОБКА МЕТОДИКИ ПРОЄКТУВАННЯ HMI НА БАЗІ SCADA TRACEMODE

Для початку створення проєкту треба зібрати його за схемою, яка зображена на рис. 3.1

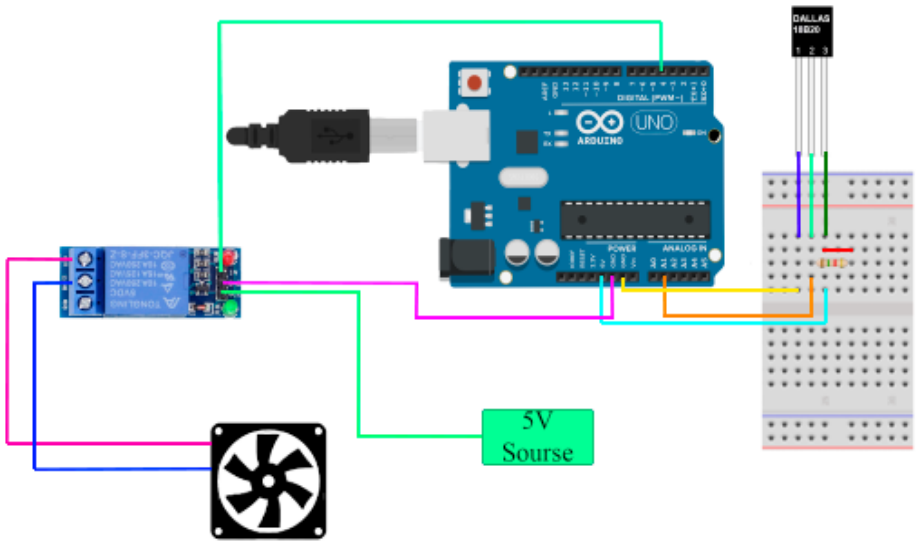

Рисунок 3.1 – Схема підключення

Після цього треба підключити пристрій до ком'ютеру і скористатись програмою FLProg для програмування контролеру. Вона запропонує дві мови на вибір FBD та LD, а також запропонє вибрати контролер. Згідно пунктів 2.1 та 2.2 обираємо мову FBD та контролер Arduino Uno (рис. 3.2).

Рисунок 3.2 – Інтерфейс вибору мови програмування і мікроконтролеру

З правої частини ми можемо бачити бібліотеку функціональних блоків. При перетягуванні елементи переносяться на схему. Щоб побачити інформацію елемента треба зробити подвійний клік по цьому елементу (рис.3.3).

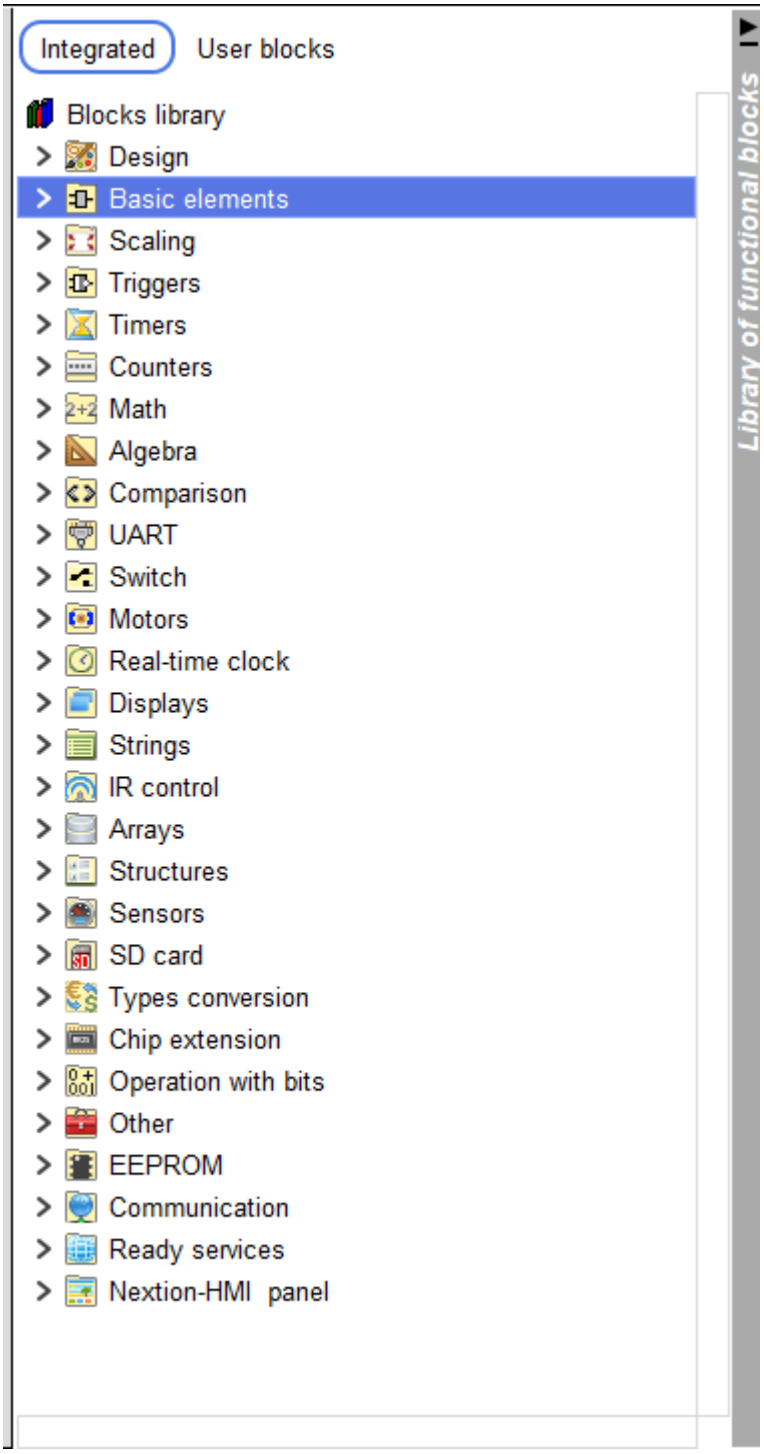

Рисунок 3.3 – Бібліотека функціональних блоків FLProg

Розробляємо модель терморегулятору (рис. 3.4)

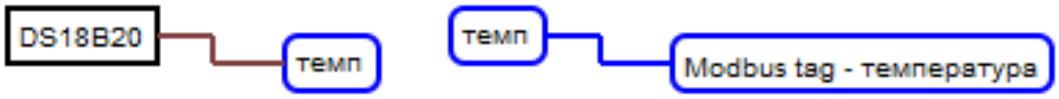

Рисунок 3.4 – Модель терморегулятора

Після цього треба приєднати до Trace Mode, за допомогою OPC-серверу. Додаємо у дерево проєкту протокол Modbus (рис. 3.5).

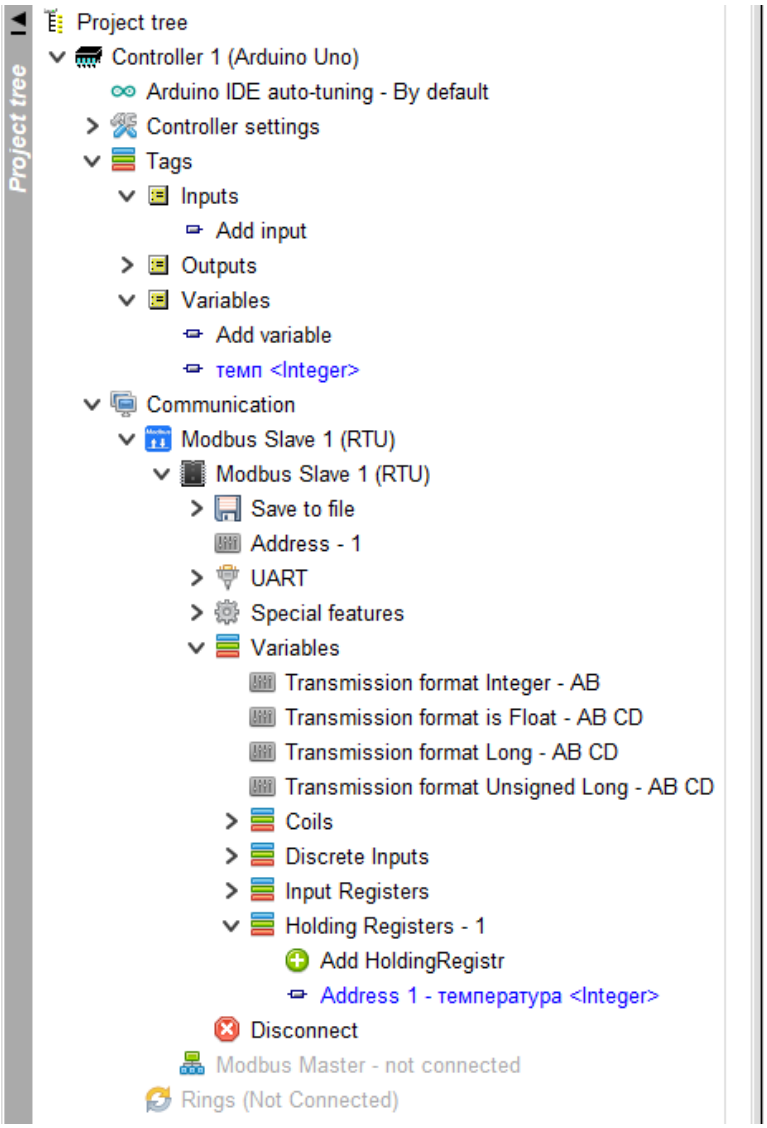

Рисунок 3.5 – Налаштування з’єднання схеми із сервером.

Залишається завантажити скетч (код програми) на мікроконтролер (рис. 3.6). Лістинг програми приведено у додатку А.

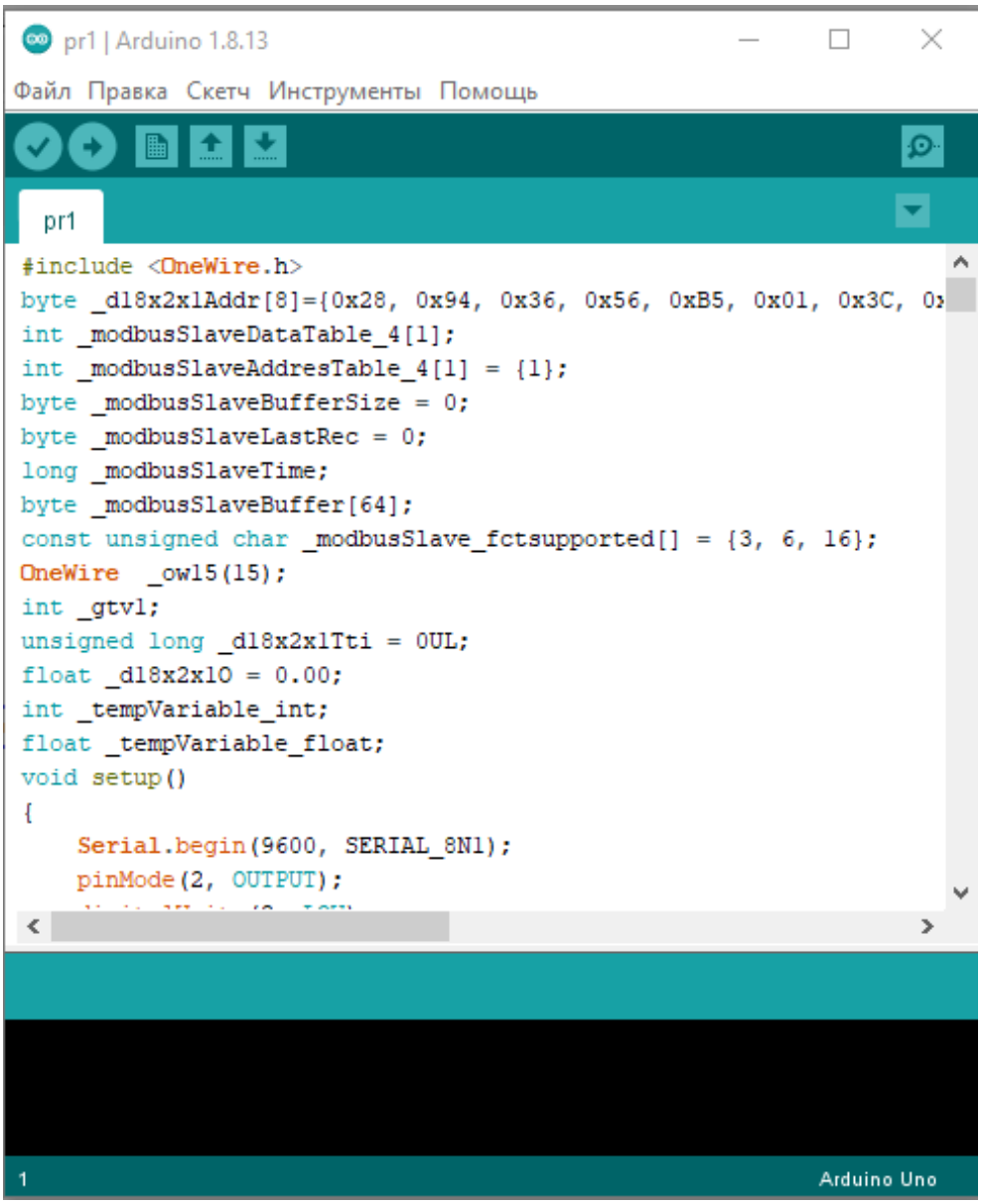

Рисунок 3.6 – Завантаження коду на мікроконтролер (частина коду)

На цьому робота в FLProg закінчена і потрібно перейти у Trace Mode для з’єднання контролеру із цією програмою і створення операторського інтерфейсу для подальшого контролю за віртуальним підприємством. Для початку створюємо новий простий проєкт (рис. 3.7).

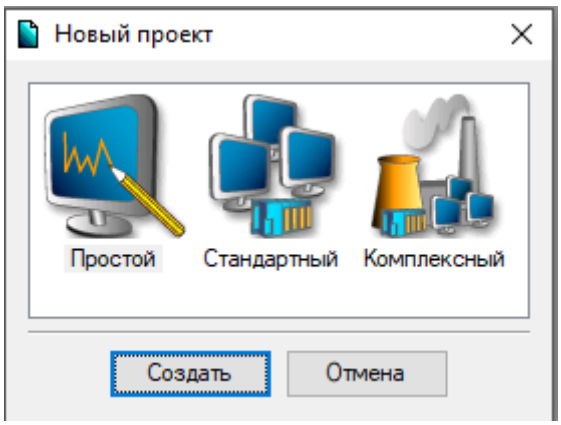

Рисунок 3.7 – Створення нового проєкту Trace Mode

У вкладці “Джерела/Приймачі” за допомогою right click створюємо групу OPC в якій такими ж маніпуляціями треба створити компонент “OPC_Сервер_1”, саме через нього до Trace Mode буде потрапляти інформація з датчику температури і ми зможемо бачити показники на графічному інтерфейсі. Потім за допомогою перетаскування компонент “OPC_Сервер_1” треба перенести у компонент “Канали” вузла системи (рис. 3.8).

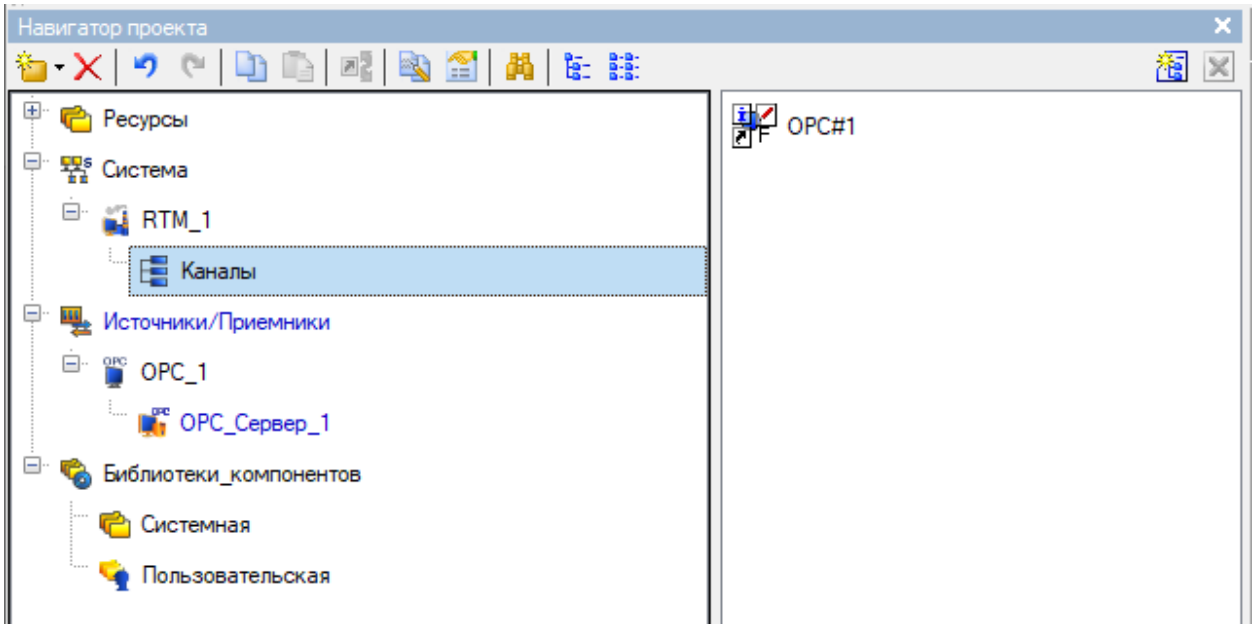

Рисунок 3.8. – Основа проєкту у Trace Mode.

Для моніторингу процесів необхідно створити інтерфейс користувача за допомогою внутрішніх засобів програми (рис. 3.9).

Рисунок 3.9 – Засоби створення інтерфейсу користувача

За допомогою засобу “Текст” створюємо дві рамки, одну з яких підписуємо “Температура” щоб позначити що саме цей показник відображає, в інший за допомогою перетаскування переносимо компонент “OPC_Сервер_1” і тепер при включенні програми на ньому буде температуру з датчика у режимі реального часу.

Відображення температури за допомогою засобу “Текст” є примітивним і тому не може використовуватись на підприємствах, оскільки людина не може контролювати процес постійно, а данні нікуди не записуються і постійно змінюються. Задля цього є більш надійні і продумані засоби, такі як “Тренди” та “Стрілочний прилад”. “Стрілочний прилад” завжди відображає задані границі температур, чи других показників які є критично малими чи великими, він є більш розвиненим засобом, але у порівнянні з засобом “Тренди” він програє, оскільки вона не тільки відображає графік змін температур, але й може бути приєднана до баз даних чи софту типу Microsoft Exel і постійно вести облік показників, що надає змогу для упереджувати критичні ситуації шляхом втручання спеціаліста, ще до того як ситуація почалась.

На рис. 3.10 показано розроблений операторський інтерфейс у середовищі програми Trace Mode v.6.0.

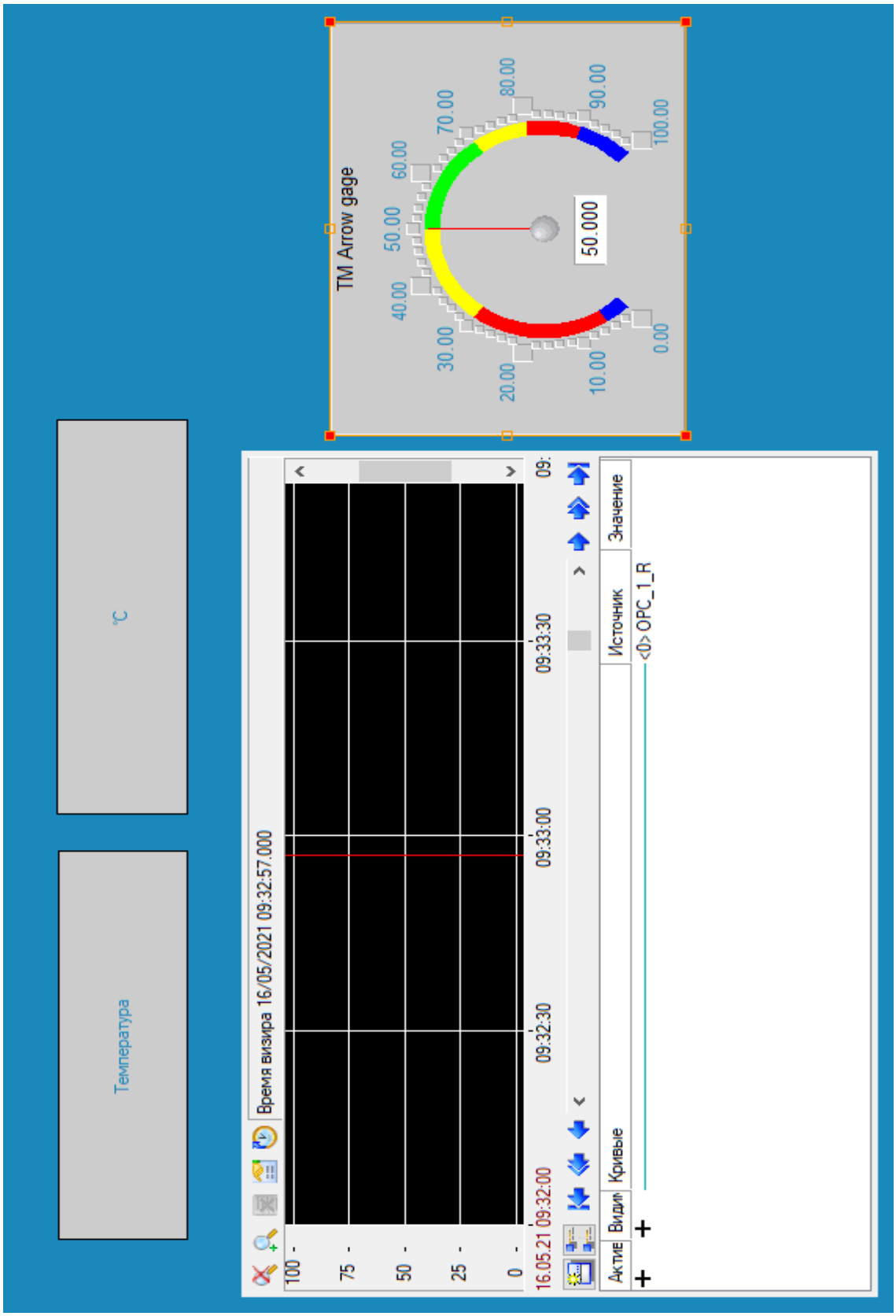

Рисунок 3.10 – Операторський інтерфейс для контролю температури

ВИСНОВКИ

В даній роботі розроблено методичне забезпечення етапу проєктування HMI в структурі віртуального виробництва.Ця методика полегшує створення аналогічного HMI віртуального виробництва на базі Trace Mode. Для верифікації методики було розроблено систему автоматичного керування і моніторингу технічного процесу на підприємстві, зокрема керування і моніторінг температури за допомогою мікроконтролера Arduino Uno, та програмного забезпечення FLProg з посиланням на Trace Mode IDE 6.

Проєкт буде актуальним для різногалузевих підприємств з великими обсягами обладнання, для яких потрібні великі потужності комп’ютерів і є неможливим хоч і більш легкий, але непотужний і малонадійний дистанційний моніторинг. Незважаючи на великий стрибок у розвитку смартфонів, ноутбуків та інших портативних гаджетів, їх потужності може не вистачити навіть для більш-менш малих підприємств, не кажучи про те, що технології бездротового зв’язку все ще неможливо представити без великих затримок та значних втрат у швидкості передачі пакетів даних, що може призвести до критичних ситуацій. Тому розробка HMI для SCADA із дротовим зв’язком для багатопараметричного керування в структурі автоматизованого виробництва є актуальною задачею.

Робота пройшла апробацію на конференції Тиждень науки 2021. Був розроблений та досліджений дослідний зразок. Загальний вигляд дослідного зразку наведений у додатку Б.

ПЕРЕЛІК ПОСИЛАНЬ

1. **Бобух, А. О.** Автоматизовані системи керування технологічними процесами [Текст]: навч. посібник / А. О. Бобух ; ХНАМГ. – Харків: ХНАМГ, 2006. – 65 с .

2. Автоматизація виробничих процесів [Текст] : підручник / І. В. Ельперін, О. М. Пупена, В. М. Сідлецький, С. М. Швед ; Нац. ун−т харч. технол. − 2−ге вид., випр. − К. : Ліра−К, 2015. − 378 с

3. Промислові мережі та інтеграційні технології в автоматизованих системах [Текст] : навч. посіб. / О. М. Пупена, І. В. Ельперін, Н. М. Луцька, А. П. Ладанюк. — К. : Ліра−К, 2011. − 552 с.

4. Certifications: Industrial control systems. [Electronic resource]. – Access mode: https://www.giac.org/certifications/ics

5. **Conte, D. L.** Designing and Developing Virtual and Real Testbeds for Industrial Control Systems Education and Training [Text]: monograph. / D.L. Conte; Eskom Peaking Engineering Head Office. – New Jersey: 2019. – 94 с.

6. **Паламар, М.В.** Проєктування комп’ютеризованих вимірювальних систем і комплексів. [Текст] / Михайло Паламар, Михайло Стрембіцький, Андрій Паламар; Тернопільський національний технічний університет імені Івана Пулюя, 2018. – 150 с.

7. **Schmutz, W. F.** Development Of The Control And Automation System For Mini Hydro Turbines : monograph. / W.F. Schmutz; ЕРЕНО – Cape Town : 1995. – 122 с.

8. SCADA системи [Електронний ресурс]. – Режим доступу: http://www.scadasystems.net/

9. SCADABR. [Electronic resource]. – Access mode: http://www.scadabr.com.br/

10. Procedure for substantiated development of measures to design secure software for automated process control systems. [Electronic resource]. – Access mode: https://ieeexplore.ieee.org/abstract/document/7491660

11. Интегрированные системы проєктирования и управления: SCADA-системы : учебное пособие / И. А. Елизаров, А. А. Третьяков, А. Н. Пчелинцев и др. – Тамбов : Изд-во ФГБОУ ВПО «ТГТУ», 2015. – 160 с. – 400 экз.

12. Programmable Controllers. [Electronic resource]. – Access mode: http://velocio.net/

13. **Samiec, T** Comparison of communication nets and programming methods fo plc simatic: тези / T. Samiec ; BUOT. – BRNO: BUOT, 2010. – 53 с.

14. ARDUINO. Arduino Uno SMD. [Електронний ресурс]. – Режим доступу: http://arduino.cc/en/Main/arduinoBoardUno

15. Датчик температури DS18B20. [Електронний ресурс]. – Режим доступу: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

16. Temperature sensor. [Electronic resource]. – Access mode: https:///www.automationdirect.com/adc/shopping/catalog/sensorszencoders/temperaturesensorsatransmitters

17. Як вибрати SCADA систему [Електронний ресурс]. – Режим доступу: http://readonline.com.ua/items/47871-yak-vibratiscada-sistemu/[ukr.]

Додаток А

Лістинг програми (розділ 3)

```
#include <OneWire.h>
byte _dl8x2xlAddr[8]={0x28, 0x94, 0x36, 0x56, 0xB5, 0x01, 0x3C, 0xEA};
int _modbusSlaveDataTable_4[1];
int _modbusSlaveAddresTable_4[1] = {1};
byte _modbusSlaveBufferSize = 0;
byte _modbusSlaveLastRec = 0;
long _modbusSlaveTime;
byte _modbusSlaveBuffer[64];
const unsigned char _modbusSlave_fctsupported[] = {3, 6, 16};
OneWire  _owl5(15);
int _gtvl;
unsigned long _dl8x2xlTti = 0UL;
float _dl8x2xlO = 0.00;
int _tempVariable_int;
float _tempVariable_float;
void setup()
{
    Serial.begin(9600, SERIAL_8N1);
    pinMode(2, OUTPUT);
    digitalWrite(2, LOW);
}
void loop()
{
    _modbusSlavePoll();
    //Board:1
    _tempVariable_int = _gtvl;
    _modbusSlaveDataTable_4[0] = _tempVariable_int;
    if(_isTimer(_dl8x2xlTti, 1000))
    {
        _dl8x2xlTti = millis();
        _tempVariable_float =  _readDS18_owl5(_dl8x2xlAddr, 0);
        if (_tempVariable_float < 500)
        {
            _dl8x2xlO = _tempVariable_float;
        }
    }
    _gtvl = (_dl8x2xlO);
}
bool _isTimer(unsigned long startTime, unsigned long period)
{
    unsigned long currentTime;
    currentTime = millis();
    if (currentTime>= startTime)
    {
        return (currentTime>=(startTime + period));
    }
     else
    {
        return (currentTime >=(4294967295-startTime+period));
    }
```

```
int modbusCalcCRC(byte length, byte bufferArray[])
{
    unsigned int temp, temp2, flag;
    temp = 0xFFFF;
    for (unsigned char i = 0; i < length; i++)
    {
        temp = temp ^ bufferArray[i];
        for (unsigned char j = 1; j <= 8; j++)
        {
            flag = temp & 0x0001;
            temp >>= 1;
            if (flag)   temp ^= 0xA001;
        }
    }
    temp2 = temp >> 8;
    temp = (temp << 8) | temp2;
    temp &= 0xFFFF;
    return temp;
}
byte _modbusSlavePoll()
{
    byte  avalibleBytes = Serial.available();
    if (avalibleBytes == 0) return 0;
    if (avalibleBytes != _modbusSlaveLastRec)
    {
        _modbusSlaveLastRec = avalibleBytes;
        _modbusSlaveTime = millis();
        return 0;
    }
    if (!(_isTimer(_modbusSlaveTime, 5))) return 0;
    _modbusSlaveLastRec = 0;
    byte state = _modbusGetSlaveRxBuffer();
    if (state < 7)
    {
        return state;
    }
    if ((_modbusSlaveBuffer[0] != 1) && (_modbusSlaveBuffer[0] != 0)) return 0;
    byte exception =  _modbusValidateRequest();
    if (exception > 0)
    {
        if (exception != 255)
        {
            _modbusSlaveBuildException(exception);
            _modbusSlaveSendTxBuffer();
        }
        return exception;
    }
    switch (_modbusSlaveBuffer[1])
    {
        case 3 :
```

```
            break;
        }
        return 0; // OK, no exception code thrown
}
bool checkModbusAddres(int addr, byte table)
{
    return (!((modbusSlaveIndexForAddres(addr,table)) == -1));
}
int modbusSlaveIndexForAddres(int addr, byte table)
{
    int tableSize = 0;
    switch (table)
    {
        case 4:
        tableSize = 1;
        break;
    }
    for (byte i = 0; i < tableSize; i++)
    {
        if((modbusSlaveAddresFromIndex(i,table)) == addr)
        {
            return  i;
        }
    }
    return -1;
}
int modbusSlaveAddresFromIndex(byte index, byte table)
{
    switch (table)
    {
        case 4:
        return _modbusSlaveAddresTable_4[index];
        break;
    }
    return -1;
}
bool checkModbusRange(int startAddr, int addrNumber, byte table)
{
    for (int i=0; i < addrNumber; i++)
    {
        if(!(checkModbusAddres((startAddr+i),table)))
        {
            return false;
        }
    }
    return true;
}
void  _modbusSlaveBuildException(byte exception)
{
    byte func = _modbusSlaveBuffer[1];
```

```
    _modbusSlaveBuffer[0] = 1;
    _modbusSlaveBuffer[1] = func + 0x80;
    _modbusSlaveBuffer[ 2 ] = exception;
    _modbusSlaveBufferSize = 3;
}
void _modbusSlaveSendTxBuffer()
{
    if(_modbusSlaveBuffer[0] == 0)
    {
        return;
    }
    byte i = 0;
    int crc = modbusCalcCRC(_modbusSlaveBufferSize,_modbusSlaveBuffer);
    _modbusSlaveBuffer[ _modbusSlaveBufferSize ] = crc >> 8;
    _modbusSlaveBufferSize++;
    _modbusSlaveBuffer[ _modbusSlaveBufferSize ] = crc & 0x00ff;
    _modbusSlaveBufferSize++;
    UCSR0A=UCSR0A |(1 << TXC0);
    digitalWrite(2, HIGH);
    delay (5);
    Serial.write(_modbusSlaveBuffer, _modbusSlaveBufferSize);
    while (!(UCSR0A & (1 << TXC0)));
    delay (5);
    digitalWrite(2, LOW);
    Serial.flush();
    _modbusSlaveBufferSize = 0;
}
byte _modbusGetSlaveRxBuffer()
{
    boolean bBuffOverflow = false;
    digitalWrite(2, LOW);
    _modbusSlaveBufferSize = 0;
    while (Serial.available())
    {
        _modbusSlaveBuffer[ _modbusSlaveBufferSize ] = Serial.read();
        _modbusSlaveBufferSize ++;
        if (_modbusSlaveBufferSize >= 64) bBuffOverflow = true;
    }
    if (bBuffOverflow)
    {
        return -3;
    }
    return _modbusSlaveBufferSize;
}
byte process_modbus_FC3(byte table)
{
    int startAddr = word(_modbusSlaveBuffer[2], _modbusSlaveBuffer[3]);
    int byteRegsno = word(_modbusSlaveBuffer[4], _modbusSlaveBuffer[5]);
    int i;
    int value;
```

```
    byte index;
    _modbusSlaveBuffer[ 2 ]  = byteRegsno * 2;
    _modbusSlaveBufferSize = 3;
    for (i = startAddr; i < startAddr + byteRegsno; i++)
    {
        index = modbusSlaveIndexForAddres(i, table);
        if (table == 4)
        {
            value = _modbusSlaveDataTable_4[index];
        }
        _modbusSlaveBuffer[ _modbusSlaveBufferSize ] = highByte(value);
        _modbusSlaveBufferSize++;
        _modbusSlaveBuffer[ _modbusSlaveBufferSize ] = lowByte(value);
        _modbusSlaveBufferSize++;
    }
    _modbusSlaveSendTxBuffer();
    return _modbusSlaveBufferSize + 2;
}
byte process_modbus_FC6()
{
    int addres = word(_modbusSlaveBuffer[2], _modbusSlaveBuffer[3]);
    int index;
    index = modbusSlaveIndexForAddres(addres, 4);
    _modbusSlaveDataTable_4[index] =word(_modbusSlaveBuffer[4], _modbusSlaveBuffer[5]);
    _modbusSlaveBufferSize = 6;
    _modbusSlaveSendTxBuffer();
    return _modbusSlaveBufferSize + 2;
}
byte process_modbus_FC16()
{
    byte func = _modbusSlaveBuffer[1];
    int startAddr = _modbusSlaveBuffer[2] << 8 | _modbusSlaveBuffer[3];
    int byteRegsno = _modbusSlaveBuffer[4] << 8 | _modbusSlaveBuffer[5];
    int i;
    int index;
    _modbusSlaveBuffer[4]   = 0;
    _modbusSlaveBuffer[5]   = byteRegsno;
    _modbusSlaveBufferSize = 6;
    for (i = 0; i < byteRegsno; i++)
    {
        index = modbusSlaveIndexForAddres((startAddr+i), 4);
        _modbusSlaveDataTable_4[index] =word(_modbusSlaveBuffer[ 7 + i * 2 ],  _modbusSlaveBuffer[8 + i * 2 ]);
    }
    _modbusSlaveSendTxBuffer();
    return _modbusSlaveBufferSize +2;
}
float _convertDS18x2xData(byte type_s, byte data[12])
{
    int16_t raw = (data[1] << 8) | data[0];
    if (type_s)
```

```
    {
        raw = raw << 3;
        if (data[7] == 0x10)
        {
            raw = (raw & 0xFFF0) + 12 - data[6];
        }
    }
    else
    {
        byte cfg = (data[4] & 0x60);
        if (cfg == 0x00)raw = raw & ~7;
        else if(cfg == 0x20)raw = raw & ~3;
        else if(cfg == 0x40) raw = raw & ~1;
    }
    return  (float)raw / 16.0;
}
float _readDS18_ow15(byte addr[8], byte type_s)
{
    byte data[12];
    byte i;
    _ow15.reset();
    _ow15.select(addr);
    _ow15.write(0xBE);
    for (i = 0; i < 9; i++)
    {
         data[i] = _ow15.read();
    }
    _ow15.reset();
    _ow15.select(addr);
    _ow15.write(0x44);
    if (_ow15.crc8(data, 8) != data[8])
    {
        return 501;
    }
    return _convertDS18x2xData(type_s, data);
}
```

Додаток Б

Загальний вигляд дослідного зразку