

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки, Факультет радіoeлектроніки та телекомунікацій

(повне найменування інституту, факультету)

Кафедра мікро- та нанoeлектроніки

(повне найменування кафедри)

Пояснювальна записка
до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему Автоматизована система із HMI на базі міні - SCADA

Виконав: студент IV курсу, групи РТ-317

Спеціальності 153 Мікро- та наносистемна техніка

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Мікро- та нанoeлектронні прилади і пристрої

Довгаль А.О.

(прізвище та ініціали)

Керівник Василенко О.В.

(прізвище та ініціали)

Рецензент Загаєвська В.І.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, 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. Сніжної Г.В.

« _____ » _____ 20__ року

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

Довгаля Артура Олексійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Автоматизована система із HMI на базі міні - SCADA

керівник проєкту (роботи) Василенко Ольга Валентинівна, к.т.н.
 доцент _____

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від « 17 » травня 2021 року № 190

2. Строк подання студентом проєкту (роботи) 25.05.2021

3. Вихідні дані до проєкту (роботи) Однопараметрична САР температури зі зворотним зв'язком з регулюванням від 18°C до 30°C; віддалений моніторинг бездротової технології для системи Android 5.1+

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Визначити етапи та види забезпечення системи автоматизації технологічних процесів, визначити вразливості та застереження їхніх складових; на основі техніко-економічних показників обрати апаратне та програмне забезпечення для реалізації мобільної системи моніторингу та керування температурою із людино - машинним інтерфейсом на базі міні-SCADA. Спроектувати та протестувати дослідний 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, 2	Василенко О.В., доц. каф. МІНЕ		
Нормоконтроль	Коротун А.В., доц. каф. 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 – 5.05.	Виконав
4	Вибір апаратного та програмного забезпечення	5.05 – 6.05.	Виконав
5	Параметризація системи	6.05 – 8.05.	Виконав
6	Розробка дослідного зразку та експерименти	8.05 – 20.05.	Виконав
7	Оформлення пояснювальної записки	20.05 – 22.05.	Виконав
8	Підготовка презентації	22.05 – 24.05.	Виконав

Студент

_____ Довгаль А.О.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

_____ Василенко О. В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

ПЗ: 45 с., 18 рис., 2 додатка, 17 джерел.

Мета роботи - спроектувати автоматизовану систему із НМІ на базі міні-SCADA.

Методи дослідження - лінгвістичні, технічні та інформаційні.

Методи проектування - програмні.

Дипломна робота складається з двох розділів.

В першому розділі було проведено інформаційне дослідження щодо мети, засобів та способів автоматизації технологічних процесів.

У другому розділі була розроблена система моніторингу та керування температурою на базі міні-SCADA.

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

ЗМІСТ

Вступ.....	7
ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ.....	6
1. АВТОМАТИЗАЦІЯ ТЕХНОЛОГІЧНОГО ПРОЦЕСУ.....	8
1.1 Підходи до автоматизації технологічних процесів.....	8
1.2 Класифікація систем керування та їх характеристики.....	12
1.3 Забезпечення автоматизованої системи керування технологічним процесом.....	14
1.4 Автоматичні та автоматизовані системи керування.....	16
1.5 Людино-машинний інтерфейс	18
1.6 SCADA системи.....	19
1.7 Типи мобільних додатків APCs і пов'язані з ними ризики	23
1.7.1 Пульт керування	23
1.7.2 Клієнт для OPC / MES або система архівації.....	24
1.7.3 Загрози для різних типів додатків.....	24
2. ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ІЗ НМІ НА БАЗІ МІНІ - SCADA	26
2.1 Розробка концептуальної моделі терморегулятора.....	26
2.2 Обґрунтування вибору апаратного забезпечення	27
2.2.1 Вибір мікроконтролера	27
2.2.2 Вибір датчика температури.....	31
2.3 Обґрунтування вибору програмного забезпечення	34
2.3.1 Вибір програми візуального програмування контролерів .	34
2.3.2 Вибір ПЗ для операційної системи ANDROID.....	38
2.4 Проєктування дистанційного терморегулятора	40
ВИСНОВКИ.....	44
ПЕРЕЛІК ПОСИЛАНЬ.....	45
ДОДАТОК А.....	47
ДОДАТОК Б	63

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ

APCS (Automated Process Control System) – автоматизована система керування технологічним процесом

ACS (Automated Control System) – автоматизована система керування

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

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

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

MIS (Manufacturing Information Systems) – виробничі інформаційні системи

MES (Manufacturing Execution Systems) – виробничі виконавчі системи

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

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

ТП – технологічний процес

ПЗ – програмне забезпечення

ОС – операційна система

ЕОМ – електронно-обчислювальна машина

САК – система автоматичного керування

ВСТУП

Сьогодні операторам потрібно виконувати все більше операцій і розглядати все більше даних в ході прийняття рішень. Якщо людино-машинний інтерфейс (англ. Human-Machine Interface, HMI), розроблений на основі аналізу кращих варіантів його практичного застосування в конкретній галузі і з урахуванням певних базових принципів, то він може поліпшити і спростити процедуру обробки інформації. При цьому оператори зможуть швидше і точніше виконувати свої обов'язки в частині контролю та поточних завдань в технологічних і виробничих процесах.

Автоматизація технологічного процесу – сукупність способів та інструментів, призначених для реалізації системи або структур, які можуть допомогти керувати самим ТП без участі робітника, чи залишення за людиною можливості прийняття найбільш важливих та відповідальних рішень.

Тенденції в галузі автоматизації ведуть до розробки HMI нового типу, спрямованого на підвищення безпеки, виконання технологічних і виробничих процесів і безпосереднє забезпечення функціонування самих виробничих об'єктів.

Представляє інтерес розробка HMI на базі SCADA для мобільних застосувань. Це дозволить уникнути обмеження традиційних SCADA, які отримують дані по наземному зв'язку. Однак, необхідно оцінити ризики бездротового зв'язку та вразливості подібних систем.

Для проєктування системи згідно вихідних даних (технічної пропозиції) необхідно проаналізувати види забезпечення автоматизованих систем, та виходячи з критеріїв оптимальності, підібрати засоби для реалізації.

Завдання даної бакалаврської роботи полягає у тому щоб розробити автоматизовану систему моніторингу та керування температурою у приміщенні із людино-машинним інтерфейсом.

1 АВТОМАТИЗАЦІЯ ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ

1.1 Підходи до автоматизації технологічних процесів

Автоматизація технологічного процесу – сукупність способів та інструментів, призначених для реалізації системи або структур, які можуть допомогти керувати самим ТП без участі робітника, чи залишення за людиною можливості прийняття найбільш важливих та відповідальних рішень [1, 2].

Головна мета автоматизації ТП – це перерозподіл інформаційних, енергетичних та матеріальних потоків відповідно до визнаних стандартів керування.

Головною метою автоматизації ТП є:

- поліпшити безпеку;
- поліпшити екологічність;
- поліпшити ефективність;
- підвищити ефективність виробничого процесу.

Досягнення цілей досягається вирішенням таких завдань:

- покращення якості регулювання;
- покращення коефіцієнта готовності обладнання;
- вдосконалення ергономіки операторів процесів;
- забезпечити достовірності інформації про склад матеріалів, що використовуються у виробництві (в тому числі за допомогою керування каталогами);
- зберігання інформації про технічні процеси та надзвичайні ситуації.

Автоматизація – одна із галузей ТП, використання саморегулюючих технічних засобів, економіко-математичних методів та структур керування, звільняє людей від процесу придбання, перетворення, передачі та використання енергії, матеріалів або інформації, тим самим значно знижуючи ступінь участі або трудомісткості виконуваних операцій. Потрібне додаткове використання датчиків, пристроїв введення, пристроїв керування

(контролерів), виконавчих механізмів, пристроїв виведення, що використовують електронні прилади та методи обчислення, іноді копіюючи нервові та психологічні функції людини. Поняття "автоматизація" використовується разом із терміном "автоматичний", що підкреслює відносно високу ступінь участі людини у процесі [1, 2].

Автоматизуються:

- процес виробництва;
- проектування;
- організація, планування та керування;
- наукове дослідження.
- бізнес-процес.

Основною метою автоматизації являється покращення індексу продуктивності праці, та підвищення якості продукції, оптимізація керування, усунення людини від виробництва яке шкідливе для здоров'я, та підвищення надійності й точності промислового виробництва, поліпшення конвертованості [1, 2].

Автоматизація потребує комплексного та системного підходу до розв'язання конкретних задач, тому їх вирішення перед автоматизацією зазвичай іменуються системами, наприклад:

- система автоматичного керування (САК) [1, 2];
- система автоматизації проєктних робіт (САПР) [1, 2];
- автоматизована система керування технологічним процесом (APCS) [1, 2].

Порівняно з попереднім етапом розвитку технологій, автоматизація має багато переваг та недоліків.

До основних переваг належать:

- заміна працівників в задачах, що включають в себе тяжку фізичну або монотонна працю;
- заміна персоналу під час виконання задач в небезпечних умовах (тобто: пожежі, виверження вулканів, стихійних лих, тощо);

- виконання завдань, що перевищують людські можливості з точки зору ваги, швидкості, витривалості, тощо;
- вдосконалена економіка. Автоматизація може покращити економічне становище підприємств, суспільства або більшості людей.

Головними недоліками автоматизації є:

- зростання кількості звільнень людей, оскільки вони замінили робочі місця машинами що призводить до безробіття;
- технічні обмеження;
- загрози / уразливості безпеки;
- у більшості випадків надто велика вартість.

Розглянемо на рис. 1.1 мету та цілі ACS.

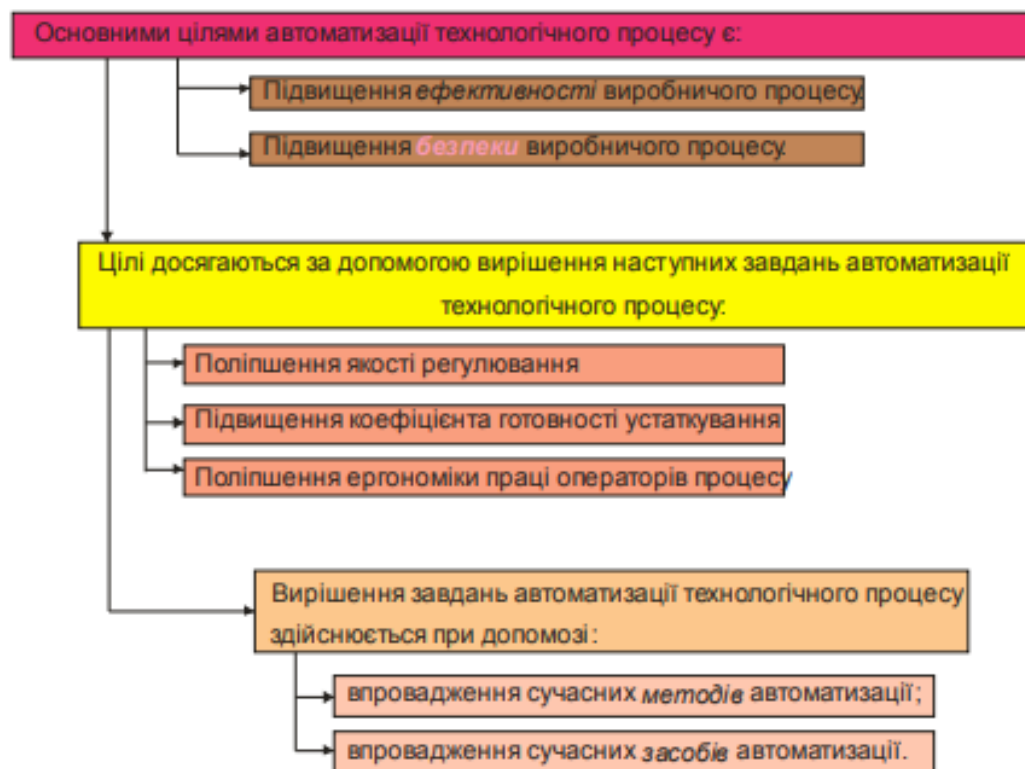


Рисунок 1.1 – Мета та цілі ACS

ТОК - технологічний об'єкт керування - комплекс технологічного встаткування й реалізованого на ньому ТП [1, 2].

ACS - автоматизована система керування - це набір апаратних та програмних засобів, призначених для керування процесами виробництва та різними процесами на підприємстві. ACS використовується у різноманітних галузях виробництва, енергетиці, транспорті і т.д. На відміну від терміну "автоматизований", термін "автоматичний" наголошує на операторів персоналу, які зберігають за собою певні функції, або найбільш універсального характеру, або непридатними для автоматизації [1, 2].

Види ACS:

- APCS - вирішує проблеми керування та контролю технічних об'єктів у промисловості, енергетиці, на транспорті;

- ACS виробництва (ACS P) - розв'язує проблеми організації виробництва, включаючи основний виробничий процес, вхідну і вихідну логістику. Виконує короткострокові виробничі плани, які повинні враховувати виробничі потужності, аналіз якості виробничої продукції та моделювання виробничого процесу. Для розв'язання цих задач застосовуються MIS і MES-системи, а також LIMS-системи [1, 2].

MIS (Manufacturing Information Systems) – інформаційна система управління, розроблена спеціально для використання у виробничих умовах. Роль виробничих інформаційних систем полягає у підтримці виробничих операцій шляхом надання відповідної та своєчасної інформації для прийняття рішень на різних рівнях ієрархії компанії. Він також автоматизує та забезпечує послідовність виробничих та бізнес-процесів.

MES (Manufacturing Execution Systems) – це динамічна інформаційна система, яка керує ефективним виконанням виробничих операцій. Використовуючи поточні та точні дані, MES направляє, спрацьовує та звітує про діяльність заводу по мірі виникнення подій. Набір функцій MES керує виробничими операціями від моменту замовлення у виробництво до моменту поставки продукції. MES надає важливу інформацію про виробничу діяльність іншим людям по всій організації та ланцюгу поставок за допомогою двонаправленого зв'язку [3].

LIMS системи дозволяють ефективно керувати потоком зразків та супутніх даних для підвищення ефективності лабораторії. LIMS допомагає стандартизувати робочі процеси, тести та процедури, забезпечуючи при цьому точний контроль процесу. Прилади можуть бути інтегровані в LIMS для автоматизації збору тестових даних, гарантуючи, що вони належним чином відкалібровані та експлуатуються лише кваліфікованим персоналом [4].

Зазвичай завдяки ACS, створюється APCS.

Компонентами APCS можуть бути окремі системи автоматичного керування (САК), або це може бути автоматизовані пристрої, з'єднані в один комплекс. Як правило APCS використовує єдину систему операторського керування ТП у вигляді одного або декількох пультів керування, засоби обробки та врахування інформації про хід процесу (датчики, контролери, виконавчі механізми). Промислова мережа використовується для інформаційного зв'язку усіх підсистем [1, 2].

У зв'язку з різними методами розрізняють автоматизацію наступних ТП:

- автоматизація безперервних ТП;
- автоматизація дискретних ТП;
- автоматизація гібридних ТП.

1.2 Класифікація систем керування та їх характеристики

Класифікація APCS, має шість найважливіших характеристик, а саме: характер керованого процесу; складність керованого процесу; ступінь охоплення керованого процесу; ступінь автоматизації завдань керування; функціонально – алгоритмічна ознака; архітектурна ознака.

За характером керованого технологічного процесу розрізняють: APCS основними безперервними ТП; ACS головними безперервно-дискретними

процесами; ACS головними дискретними технологічними процесами; ACS складальними процесами в дискретному виробництві; ACS процесами виготовлення обладнання основних виробничих інструментів.

ACS складальними процесами представляють вищий рівень комплексних ACS технологічними процесами і розв'язують задачу координації в реальному часі ряду дискретних ТП або операцій [1,2].

Класифікація за складності керованого ТП базується на умовному обмеженні кількості параметрів контролю та керування процесом. Наприклад, 20, 40, 100, 800 параметрів. Ця класифікація, хоча і звичайна, може бути використана як основа для планування розробок [1,2].

Залежно від ступеня охоплення керованого ТП виділяють комплексні та локальні APCS.

Залежно від ступеня автоматизації завдань керування виділяють деякі системи які мають функцію автоматичного збору та обробки інформації; системи з автоматичним генеруванням рад людині; САК процесом, або: інформаційні, інформаційно – відповідні та керуючі.

Класифікація за функціонально - алгоритмічною ознакою визначає функції і досконалість алгоритму керування, реалізованого APCS.

Розділяють:

- 1) системи логіко-програмного керування;
- 2) системи екстремального керування;
- 3) адаптивні системи керування;
- 4) системи організаційно-технологічного керування;
- 5) системи оптимально-координаційного керування.

Класифікація за архітектурною ознакою:

- однорівневі централізовані структури на основі керуючого обчислювального комплексу, з прямим зв'язком з усіма джерелами і приймачами інформації;

- однорівневі централізовані структури з ущільненням каналів зв'язку - системи на основі одного управляючого комплексу та систем ущільнення ліній зв'язку;

- дворівневі з однієї ЕОМ - системи на основі однієї ЕОМ з частотним розподілом функцій керування на керовані регулятори і (або) програматори, і (або) локальні пости керування;

- багаторівневі (дворівневі), з багатьма ЕОМ - системи, в яких ЕОМ використовується більш, ніж на одному рівні.

1.3 Забезпечення автоматизованої системи керування технологічним процесом.

ПЗ АPCS поєднує усі частини розподіленої технічної структури в єдине ціле та має безліч переваг:

- оскільки АСS технологічного процесу поділяється на сімейства відносно невеликі та менш складні автономні підсистеми з додатковим резервуванням через мережу, можна отримати більш високу надійність;

- використання більш надійних засобів мікроелектронної обчислювальної техніки;

- велика гнучкість при композиції й модернізації апаратного та програмного забезпечення.

Більшість функцій АPCS реалізуються програмним забезпеченням, тому найважливішим частиною АPCS яке являє собою набір програм, що реалізують функції АPCS. ПЗ автоматизованої системи керування технологічного процесу поділяється на:

- загальне;
- спеціальне.

Загальне ПЗ постачається в комплекті з комп'ютерною технікою.

При створенні конкретній APCS, буде розроблено спеціальне ПЗ, включаючи програми для реалізації його інформаційних та керуючих функцій.

ПЗ створюється на основі математичного забезпечення. Математичне ПЗ - комплекс математичних алгоритмів, моделей та методів, що застосовують комп'ютерні технології для розв'язання завдань та обробки інформації.

Для реалізації інформаційних і контролюючих функцій APCS було створено спеціальне математичне забезпечення, яке включає в себе:

- алгоритм збору, обробки й подання інформації;
- алгоритми керування з математичними моделями відповідних об'єктів керування;
- алгоритми локальної автоматизації.

Всі взаємодії як усередині APCS, так і із зовнішнім середовищем є різними формами обміну інформацією, що є необхідними масивами даних, щоб APCS виконувала всі свої функції.

Правила обміну інформацією й інформацією, що циркулює в APCS, утворюють інформаційне забезпечення APCS.

Організаційна підтримка APCS це опис функцій системи, технічної та організаційної структури системи, а також інструкцій та правил для оперативного персоналу, який забезпечує конкретні функції APCS.

До операторів APCS належать з технологи-оператори, що здійснюють керування ТОК, експлуатаційного персоналу, які забезпечують нормальну роботу ACS технологічного процесу (оператори, програмісти, і робочі, що обслуговують обладнання комплексу технічних засобів).

Оператори APCS можуть працювати в контурі керування або зовні нього.

Працюючи в контурі управління, працівник виробництва виконуватиме усі функції керування чи їх частину та використовуватиме ряд технічних заходів для формування рекомендацій щодо керування ТОК.

Такий режим функціонування APCС називають інформаційно - консультаційним.

Якщо оператори працюють поза контуром управління, він задасть APCС режим роботи який здійснює контроль за його дотриманням. У цьому випадку APCС може працювати у двох режимах відповідно складу комплексу технічних засобів:

- комбінованому;
- у режимі прямого цифрового керування, при якому УВК безпосередньо впливає на виконавчі пристрої, змінюючи тим самим керування впливу на ТОК.

1.4 Автоматичні та автоматизовані системи керування

У розвитку виробництва, де переважають безперервні технологічні процеси (наприклад, нафтохімічна, нафтопереробна та металургійна) зажадало створення більш досконалих систем керування, ніж місцеві.

Завдяки створенню комп'ютера та збільшення їх обчислювальних ресурсів та надійності, створення APCС стало можливим.

Розглянемо на рис. 1.2 труднощі переходу від APCС до повністю автоматичних виробництв.

APCS - називають ACS який використовується для розробки та реалізації керуючого ефекту на ТОК відповідно до визнаних стандартів керування та є показником, який показує якість роботи ТОК та приймає конкретне значення відповідно до використовуваних керуючих впливів [1,2].

АТК - група технологічних об'єктів керування які працюють разом з ACS технологічного процесу; вони утворюють автоматизований технологічний комплекс [1,2].

Глобальна мета керування технологічними об'єктами керування за допомогою APCС складається у тому що треба підтримувати екстремальне

значення стандартів керування при виконанні за будь-яких умов, що визначають безліч допустимих значень керуючих впливів.

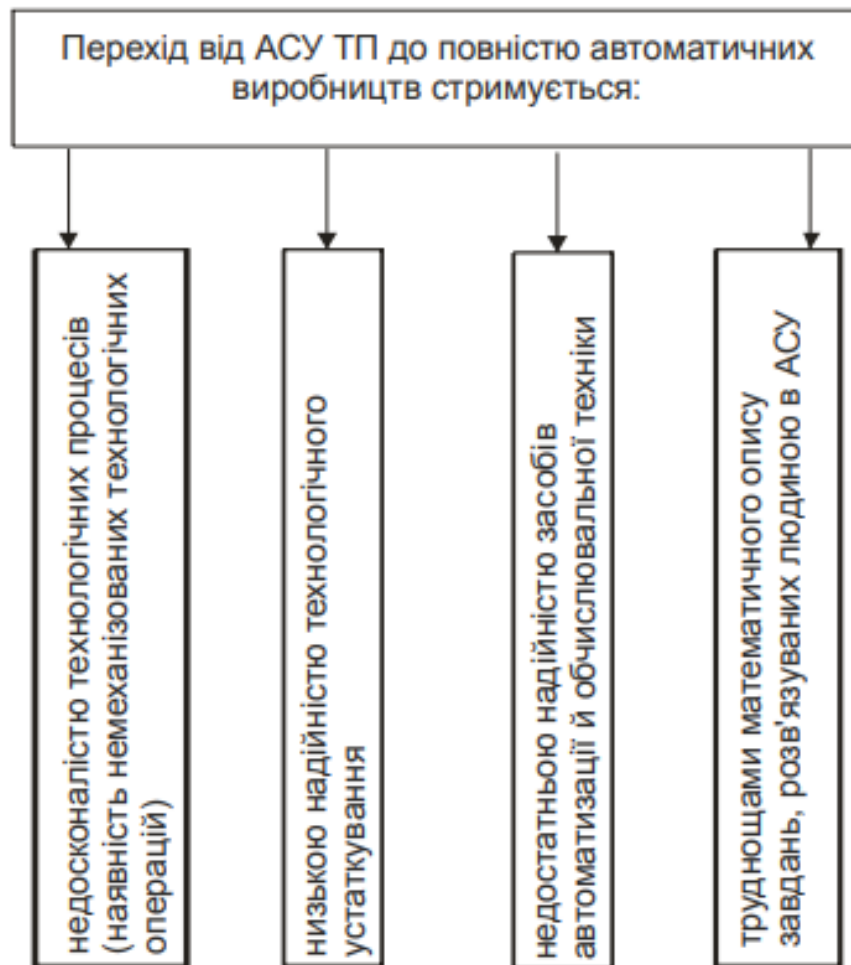


Рисунок 1.2 – Труднощі переходу від APCS до повністю автоматичних виробництв

У більшості випадків глобальні цілі поділяються на декілька приватних цілей. Для реалізації кожної з них потрібні рішення для більш простих завдань керування. Розглянемо на рис. 1.3 типову функціональну структуру APCS, де:

- 1 - основна обробка інформації;
- 2 - виявлення відхилень технологічних параметрів і параметрів стану встаткування від встановленими значеннями;
- 3 - розрахунок не вимірювальних значень та показників;

- 4 - підготовка інформації та виконання процедур обміну із відповідними та іншими ACS;
- 5 - оперативне АБО по виклику відображення й реєстрація інформації;
- 6 - визначення розумного режиму технологічного процесу;
- 7 - здійснення формування керуючих впливів, обраного режиму.

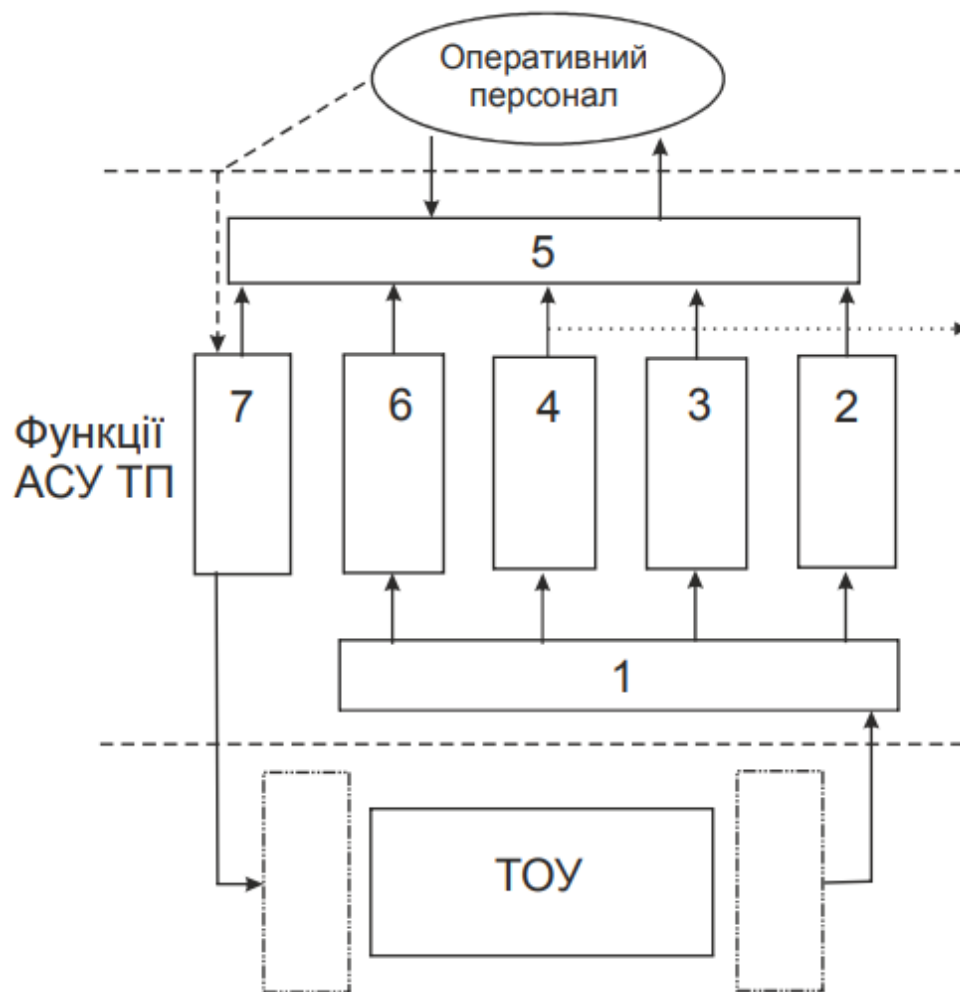


Рисунок 1.3 – Типова функціональна структура АРС

1.5 Людино-машинний інтерфейс

НМІ – це аббревіатура Human-Machine Interface - людино-машинний інтерфейс. Це широке поняття, яке охоплює інженерні рішення, що дозволяють людині-оператору взаємодіяти з керованими їм машинами. У

нашому випадку це панель на базі смартфона / планшета з ОС ANDROID яка дозволяє створювати проєкти візуалізації для пристроїв, що підтримують протокол MODBUS RTU через Bluetooth, WiFi і USB. [5]

MODBUS RTU - комунікаційний протокол, заснований на архітектурі ведучий-ведений (master-slave).

1.6 SCADA системи

SCADA - аббревіатура від Supervisory Control And Data Acquisition - диспетчерське керування і збір та обробка даних в реальному часі. Це система управління автоматизацією, яка використовується в таких галузях, як енергетика, нафта і газ, вода, енергетика та багато інших. Система має централізовану систему, яка контролює та керує цілими ділянками, починаючи від промислового заводу і закінчуючи комплексом заводів по всій країні. SCADA працює з сигналами, які обмінюються даними по каналах, щоб забезпечити користувача дистанційним управлінням будь-яким обладнанням у цій системі. Також SCADA реалізує розподілену базу даних, або базу даних тегів, яка містить в собі теги або точки на всій установці. Ці точки представляють єдине вхідне або вихідне значення, яке контролюється системою SCADA в централізованій диспетчерській кімнаті. Точки зберігаються у розподіленій базі даних мітки значення-час. Дуже поширеним є налаштування систем SCADA для отримання метаданих, таких як шляхи реєстрації програмованого логічного контролера (ПЛК) та статистика тривоги [6].

Хоча ці системи спрощують задану інфраструктуру, їх компоненти досить складні. Існує п'ять важливих складових частин системи SCADA:

- 1) людино-машинний інтерфейс (HMI);
- 2) диспетчерська система;
- 3) віддалений термінал (RTU);

- 4) програмовані логічні контролери (ПЛК);
- 5) інфраструктури зв'язку.

HMI обробляє дані з кожного тегу і надсилає їх оператору-людині, де він або вона потім може слідити або керувати системою. Диспетчерська система збирає дані які посилаються від кожного тегу, і надсилає команди або операції процесу. RTU підключають датчики і перетворюють їх сигнали в цифрові дані і надсилають їх до системи контролю, де вони можуть зберігатися в розподіленій базі даних. PLC використовуються як польові пристрої, бо вони більш універсальні та економічні, ніж RTU [6].

Нарешті, комунікаційна інфраструктура забезпечує підключення до диспетчерської системи, а потім до RTU та PLC, щоб користувач мав команду. Інфраструктура зв'язку необхідна для того щоб передавати дані з віддалених RTU / PLC, які проходять вздовж електричних мереж, водопостачання та трубопроводів. Зв'язок є абсолютно найважливішою ланкою для нормальної роботи системи SCADA; однак, незважаючи на те як добре система керує зв'язком від HMI до RTU і PLC визначає наскільки успішною може бути система SCADA. На рис. 1.4 зображено, як може виглядати базова система SCADA для даної інфраструктури [6].

Системи SCADA тепер мають доступну потужність хмарних обчислень; ці системи можуть повідомляти про точність в режимі реального часу та використовувати хмарні середовища для реалізації більш складних алгоритмів. В іншому випадку ці алгоритми не можна було б реалізувати на традиційних PLC або RTU. Навіть не будучи на заводі, працівники можуть отримати доступ до обчислювальних ресурсів, таких як мережі, сховища, сервери та засоби управління [6].

Хмарні обчислення можуть підтримуватися двома способами: система SCADA працює на місці, підключена безпосередньо до комунікаційної інфраструктури, і доставляє інформацію до хмари, або SCADA-система повністю працює в хмарній мережі та віддалено підключена до комунікаційної інфраструктури. Як би практичним не був доступ до

елементів керування на відстані, хмарні обчислення за допомогою додатків SCADA все ще дуже вразливі до кібератак. Якби система була атакована, хакери могли отримати доступ до організаційних даних та ресурсів [6].

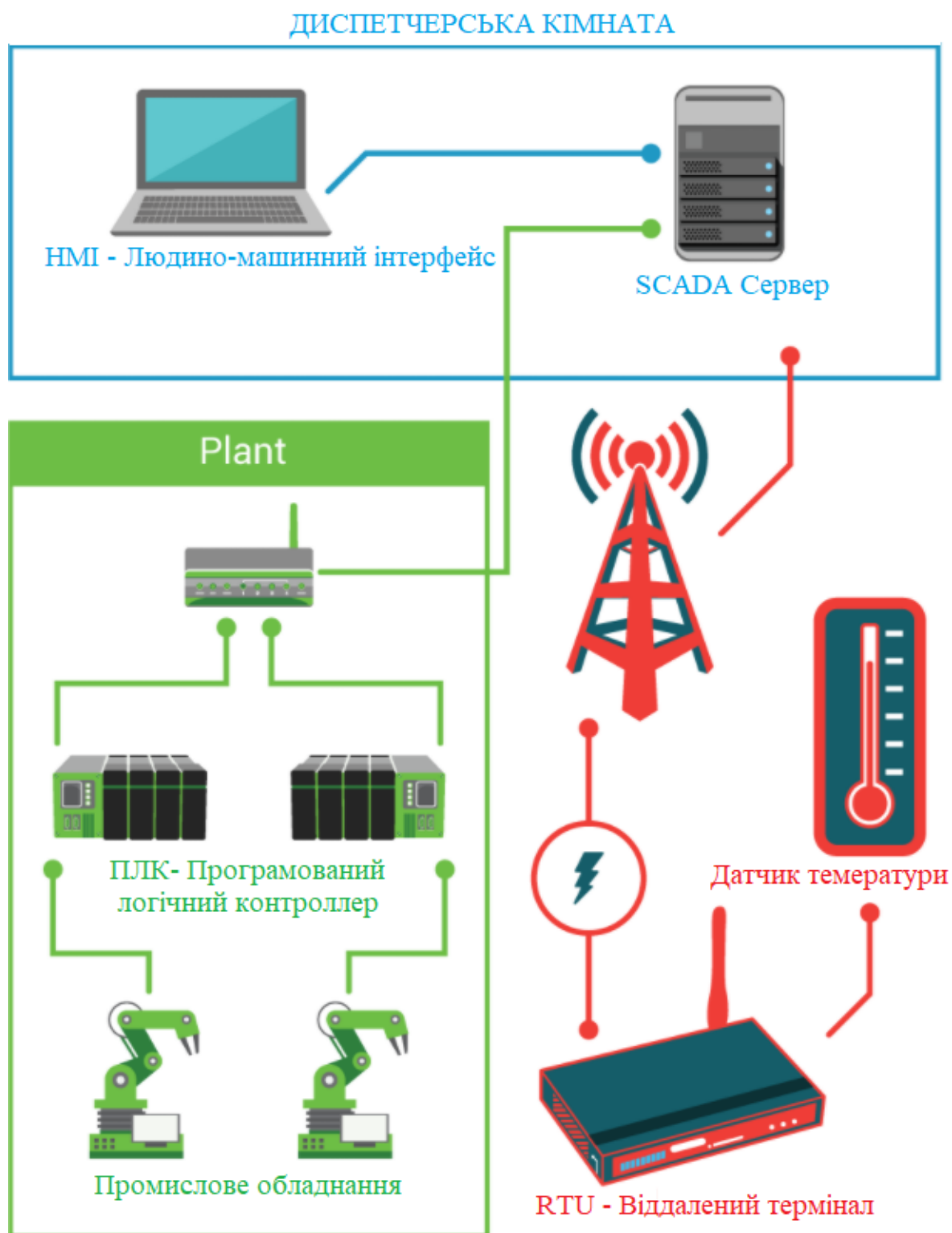


Рисунок 1.4 - Базова система SCADA

Віддаленим терміналом у нашій досліджуваній системі є мікроконтролер ESP-8266 NodeMcu v3, технічні характеристики якого наведені нижче, він здійснює обробку завдання по засобам написаної програми на мові програмування Lua, комунікаційною системою, що здійснює обмін даними між віддаленим терміналом і диспетчерським пунктом.

Програмний код може бути як написано на одному з мов програмування, так і згенерований в середовищі проектування.

Можна виділити наступні основні групи SCADA-систем:

- SCADA-системи, розроблені виробниками контролерів;
- SCADA-системи, розроблені незалежними виробниками;
- SCADA-системи складові частини систем наскрізного програмування.

Завданням виробника контролерного обладнання при розробці власної SCADA-системи є надання кінцевому користувачеві кошти для розробки додатків візуалізації при використанні контролерів цього виробника [6].

Можна виділити наступні основні риси таких систем:

- інтерфейс цих систем повторює інтерфейс засобів написання ПЗ для контролерних обладнань;
- компоненти SCADA-системи оптимізовані для роботи з даними, що отримуються від контролерних обладнань конкретного виробника;
- інтерфейси обміну даними з обладнанням інших виробників реалізовані слабо, або їх використання ускладнене.

1.7 Типи мобільних додатків APCS і пов'язані з ними ризики

1.7.1 Пульти керування

Пульт керування здійснює функції конфігурація, моніторинг, контроль виробничого процесу або його компонентів. Ролі в інфраструктурі APCS :

- конфігурація ПЛК. Налагодження або контроль компонентів APCS: ПЛК, HMI, підключених до PLC, RTU та іншого. Мобільний додаток підключається до компоненту APCS з промислової мережі (нижній і середній рівні інфраструктури). Воно здійснює контроль над станом компонента або його (пере) настройку. Можна вважати його аналогом стандартного переносного терміналу, захованим всередині Nexus або Xperia;

- клієнт для SCADA. Контроль над промисловим процесом за допомогою SCADA-клієнта або віртуального інтерфейсу для підключення до SCADA. Ці програми дозволяють інженерам-технологам підключатися з смартфона або планшета до SCADA-додатком і при необхідності контролювати виробничий процес;

- Мобільна HMI-панель. HMI-панель всередині мобільного пристрою для контролю або управління декількома промисловими компонентами. Дозволяє інженеру формувати (і навіть програмувати!) HMI-панель і підключати її компоненти власне до ПЛК або інших пристроїв через Modbus / TCP, OPC або інші протоколи та інтерфейси.

У всіх цих додатків є одна важлива особливість - з'єднання між додатком і промисловим компонентом відбувається в імовірно безпечному середовищі, десь на нижніх і середніх рівнях APCS.

В джерелах [12-17] наведені різноманітні аспекти вибору апаратного забезпечення та HMI для моніторингу на базі SCADA, згідно рекомендацій яких велася дана робота.

1.7.2 Клієнт для OPC / MES або система архівації

Це програми-архіватори дозволяють інженеру або власнику процесу читати і інтерпретувати деякі відомості з середнього і вищого рівня компонентів APCS. Дані доступні лише для читання, причому у користувача навіть немає прямого доступу до ПЛК, HMI або SCADA - тільки можливість перегляду певних змінних. У цю ж категорію ми включили мобільні клієнти для MES-додатків. Вони, як правило, використовуються, щоб читати дані з сервера архівації або агреговані дані від MES-системи. Типове додаток з цієї категорії - клієнт (браузер) для OPC. Він підключається не безпосередньо до компонентів APCS, а до сервера OPC. З'єднання зазвичай відбувається на вищих рівнях інфраструктури. Крім того, підключення може бути виконано дистанційно. Це означає, що в такому додатку повинні бути впроваджені механізми шифрування і аутентифікації.

Хоча читання агрегованих даних саме по собі не становить загрози, ця інформація може бути використана зловмисником для розуміння (зворотної розробки) промислового процесу і створення спеціалізованого сценарію атаки. Крім того, якщо додаток застосовується поза заводської мережі, з'являються нові загрози: наприклад, компрометація телефону інженера або керівника через уразливість в додатку.

1.7.3 Загрози для різних типів додатків

Пульти керування – відсутність на стороні сервера перевірки даних з точки зору промислового процесу. Через те, що серверний додаток повністю довіряє даним, які прийшли від оператора, і коректність технологічного процесу повністю залежить від дій і команд, які він відправляє, будь-які недоліки валідації даних можуть привести до серйозних наслідків. При цьому, в мобільному додатку легше помилитися в розряді числа або

випадково визначення від'ємного значення параметра. Невірна інтерпретація значень так-же може привести до відмови в обслуговуванні як серверного, так і клієнтської програми.

Компрометація даних, що зберігаються на SD-карті - Багато інших програм зберігають дані на SD-карті: логи, проекти HMI і інші дані. Відповідно, всі вони мають доступ до загальної інформації. Додаток, встановлений зі сторонніх джерел (або підроблене з офіційного магазину), може читати або модифікувати дану інформацію - наприклад, змінити інтерфейс або функціонал мобільного HMI-додатки.

Клієнт для OPC / MES або система архівації - дані додатку знаходяться вже не в захищеному контурі і передаються по мережах загального призначення (Інтернет), можлива компрометація сервера і програми за допомогою перехоплення даних, які надходять від програми до сервера. Зловмисник може отримати доступ до авторизаційної інформації.

2 ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ІЗ НМІ НА БАЗІ МІНІ - SCADA

2.1 Розробка концептуальної моделі терморегулятора

Основу приладу складає Wi-Fi модуль ESP-8266 NodeMcu v3, до якого буде підключено цифровий датчик вимірювання температури DS18B20, та двоканальний релейний модуль за допомогою якого можна буде вмикати або вимикати наш обігрівач. Wi-Fi модуль буде передавати дані з датчика температури на мобільний додаток HMI-KaScada у якому можливо стежити та задавати потрібну температуру,. На рис. 2.1 зображена структурна схема моделі терморегулятора.

Апаратне забезпечення цього проєкту представляє собою:

1. Wi-Fi модуль ESP-8266 NodeMcu v3;
2. телефон Xiaomi Mi Play з операційною системою ANDROID;
3. датчик температури DS18B20;
4. двоканальний релейний модуль;
5. нагрівальний елемент.

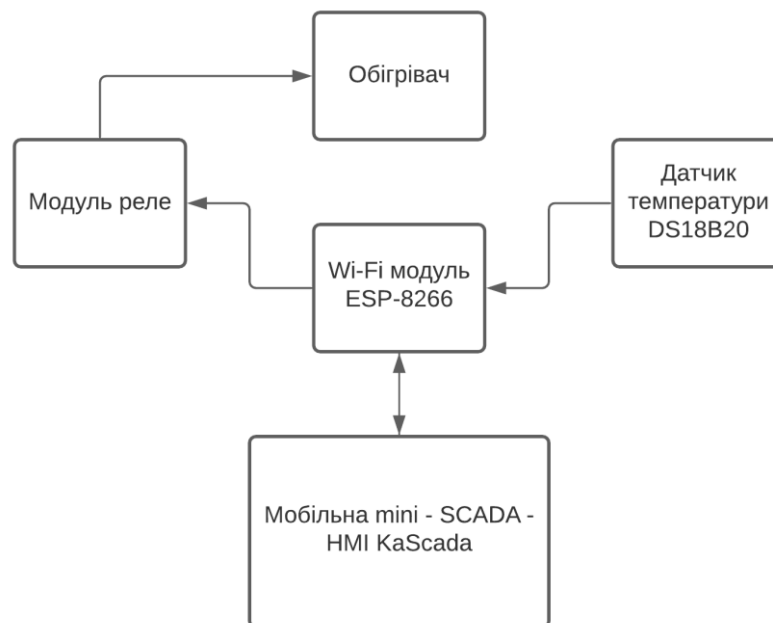


Рисунок 2.1 – Структурна схема моделі терморегулятора

Програмне забезпечення цього проєкту представляє собою:

1. FLProg - система візуального програмування контролерів;
2. HMI-KaScada – мобільний додаток для віддаленого керування різноманітними приладами.

2.2 Обґрунтування вибору апаратного забезпечення

2.2.1 Вибір мікроконтролера

В даний час промисловість пропонує широкий спектр Wi-Fi модулів, що здатні відправляти а також отримувати інформацію в локальну мережу або в інтернет.

При виборі контролера нам необхідно орієнтуватися на програмне забезпечення що ідеально підходить для програмування нашого Wi-Fi модуля.

Вибираємо Wi-Fi модуль ESP-8266 NodeMcu v3, зображений на рис. 2.2.

NodeMcu - це платформа на основі модуля ESP8266. Плата призначена для зручного управління різними схемами на відстані за допомогою передачі сигналу в локальну мережу або інтернет через Wi-Fi. Ця проста плата має досить великий спектр можливостей. Наприклад, на базі Node MCU можна створити «розумний будинок», налаштувавши управління світлом або вентиляцією через телефон, реєстрацію показників датчиків і керувати всім цим на відстані.

Характеристики NodeMcu:

- Розмір плати NodeMcu - 6*3 см. Плата досить компактна, це дозволяє використовувати її в більшій кількості проєктів. Контакти NodeMcu розташовані так, що її без проблем можна встановити в макетну плату (breadboard);

- На лицьовій частині плати гніздо Micro USB, за допомогою якого в контролер заливають скетчі або подають живлення від powerbank-а або комп'ютера;

Поруч з гніздом розташовуються дві кнопки: «Flash» і «Reset». Кнопка «Flash» використовується для налагодження, а кнопка «Reset» для перезавантаження плати [7];

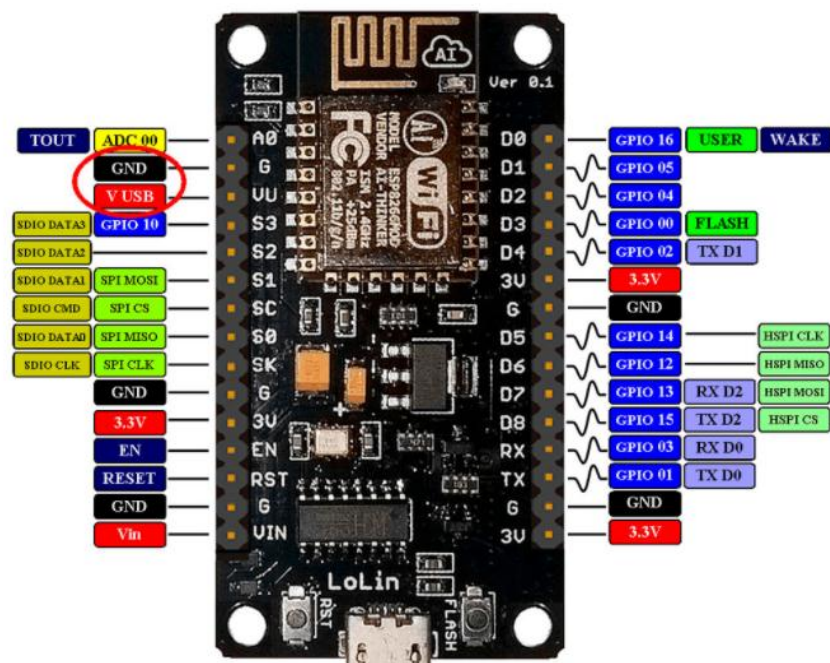


Рисунок 2.2 – Wi-Fi модуль ESP-8266 NodeMcu v3 [7]

Технічні характеристики Wi-Fi модуля зображені на рисунку 2.3

- підтримує Wi-Fi протокол 802.11 b / g / n ;
- підтримувані режими Wi-Fi - точка доступу, клієнт;
- вхідна напруга 3,7 В - 20 В;
- робоча напруга 3В-3,6;
- максимальний струм 220мА ;
- вбудований стек TCP / IP;
- діапазон робочих температур від -40С до 125С;
- 80 МГц, 32-бітний процесор;
- час пробудження і відправки пакетів 22МС;
- вбудовані TR перемикач і PLL.

Рисунок 2.3 – Технічні характеристики Wi-Fi модуля

Опис та призначення виводів ESP-8266 NodeMcu v3:

- GND – земля;
 - Vin – вивід використовується для підключення зовнішнього джерела живлення 5V. Стабілізатор AMS1117-3.3 дозволяє подавати живлення на Vin в широкому діапазоні від 5 до 10 V [7];
 - 3.3V – контакт вихідної напруги внутрішнього стабілізатора. Може бути використаний для живлення датчиків що підключаються до плати. Сумарна максимальна навантаження всіх виводів 3.3V не повинна перевищувати 300 мА [7];
 - GPIO (General Purpose Interput Output) – контакти загального призначення для введення / виведення даних. Їх можна налаштувати як входи або виходи, а також програмно призначити їх на різні функції [7];
 - RST (Reset) – вивід використовується для скидання мікроконтролера ESP8266 [7];
 - EN - при подачі на виведення сигналу високого рівня, мікроконтролер ESP8266 переходить в робочий режим, при сигналі низького рівня відповідно в режим низького споживання енергії (режим енергозбереження) [7];
 - WAKE – вивід який призначений для пробудження чіпа ESP8266 з режиму глибокого сну (deep-sleep mode) [7];
 - ADC0 / TOUT – вивід вбудованого 10-розрядного аналого-цифрового перетворювача (АЦП). Перетворені значення лежать в інтервалі 0-1023. Плата розробки NodeMCU V3 має внутрішній дільник напруги, вхідний діапазон АЦП становить 0 - 3,3 В [7];
- UART – це асинхронний послідовний інтерфейс задача якого встановити зв'язок з іншими пристроями по шині UART [7];
- SPI (Serial Peripheral Interface) – периферійний послідовний інтерфейс. NodeMCU має два SPI (SPI і HSPI) в провідному і підпорядкованому режимах [7];

- SDIO – інтерфейс безпечних цифрових входів та виходів, призначений для зв'язку з зовнішньою флеш-пам'яттю а саме SD, по послідовній шині [7];

- FLASH - Задача кнопки FLASH – підключення до землі GPIO0. Також її можна використовувати як звичайну кнопку. Якщо програмно підтягнути вивід GPIO0 за допомогою внутрішнього резистора, то поява низького рівня на цьому виводу буде означати, що кнопка натиснута [7];

- Інтерфейс I2C – послідовна асиметрична шина. I2C використовується для підключення датчиків і периферійних пристроїв. NodeMCU ESP8266 не має апаратних висновків I2C, але інтерфейс можна реалізувати програмно [7];

- PWM (Pulse-Width Modulation) – широтно-імпульсна модуляція (ШИМ) управляє потужністю методом пульсуючого включення і виключення виводу. NodeMCU підтримує програмний ШИМ на виводах, позначених на малюнку зігнутою лінією [7];

- наявність підсилювачів потужності, регуляторів, систем управління живленням [7];

Основні переваги даної плати на основі модуля ESP8266:

- на платі присутній інтерфейс UART-USB з гніздом micro-USB, що дозволяє підключати його до комп'ютера без перехідників;

- плата має виводи для всіх доступних контактів ESP8266. А це 11 портів введення та виведення загального призначення, деякі з яких мають додаткові функції;

- на даній платі встановлена прошивка, яка може інтерпретувати команди скриптові мови Lua.

Lua - це вбудована мова сценаріїв, який є маленьким, швидким і дуже потужним інструментом. За допомогою команд Lua для NodeMCU можна виконувати наступні дії:

- використання плати в якості WiFi точки доступу;
- підключатися до точки доступу WiFi;

- режим зниженого енергоспоживання;
- перенаправляти вивід;
- виконувати операції зі списком файлів у flash-пам'яті;
- управління призначеним для користувача таймером;
- управління GPIO1 висновками;
- створення веб-сервера;
- зчитувати дані на виведенні АЦП.

Можна не тільки виконувати команди Lua в терміналі, але і створювати файли в flash-пам'яті ESP8266 і викликати їх на виконання.

2.2.2 Вибір датчика температури

Для нашого проєкту нам був потрібен простий та дешевий датчик температури який працює в діапазоні від 18°C до 30°C з точністю $\pm 0.5^\circ\text{C}$. Тому було прийнято рішення використати датчик температури DS18B20 у корпусі TO-92, вартість якого складає 20 грн.

Цифровий датчик DS18B20 забезпечує вимір температури з 9...12-розрядним розширенням в градусах Цельсія, і має сигнальну функцію з енергозалежним запрограмованим користувачем верхнім та нижнім лімітом тригера. Підключення датчика DS18B20 забезпечується за допомогою шини 1-Wire, яка вимагає тільки одну лінію даних (та землі) для досягнення зв'язку з центральним мікропроцесором. Діапазон вимірюваної температури - від -55 °C до + 125 °C з точністю $\pm 0.5^\circ\text{C}$ в інтервалі від -10 °C до + 85 °C. Крім того, DS18B20 може отримувати живлення безпосередньо від лінії даних ("паразитне живлення"), усуваючи потребу в зовнішньому джерелі живлення. Схеми нормального і паразитного живлення зображені на рис. 2.4 та 2.5 відповідно [8].

Кожен датчик DS18B20 має свій унікальний 64-розрядний серійний номер, який дозволяє безліч DS18B20 працювати на одній шині. Це означає,

що одним мікропроцесором можна керувати багатьма датчиками DS18B20, які розподілені на великій площі. DS18B20 можуть бути корисними в системах екологічного контролю, включаючи HVAC, системах контролю температури в приміщеннях, обладнання, машин, системах контролю [8].

Датчик поставляється в трьох корпусах: 3-х контактний TO-92 (DS18B20+), 8-контактний SO (DS18B20Z+) і 8-контактний μ SOP (DS18B20U+). Так само, TO-92, поставляється в водонепроникному виконанні, так що можна використовувати його в бойлер, акваріумі або навіть в морозильній камері, без додаткового захисту від води або вологи [8].

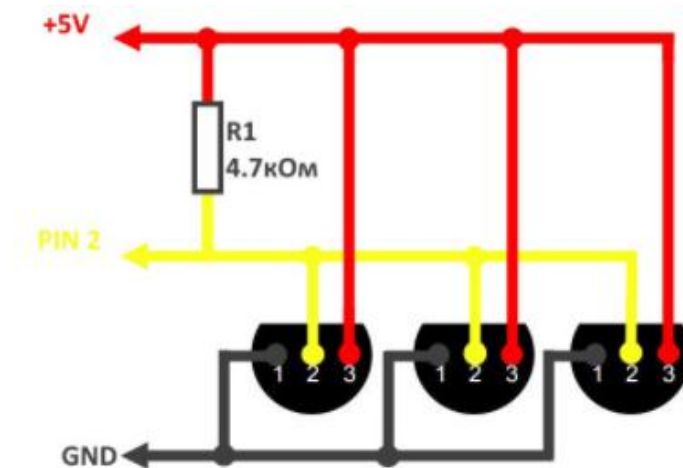


Рисунок 2.4 – Схема нормального включення DS18B20

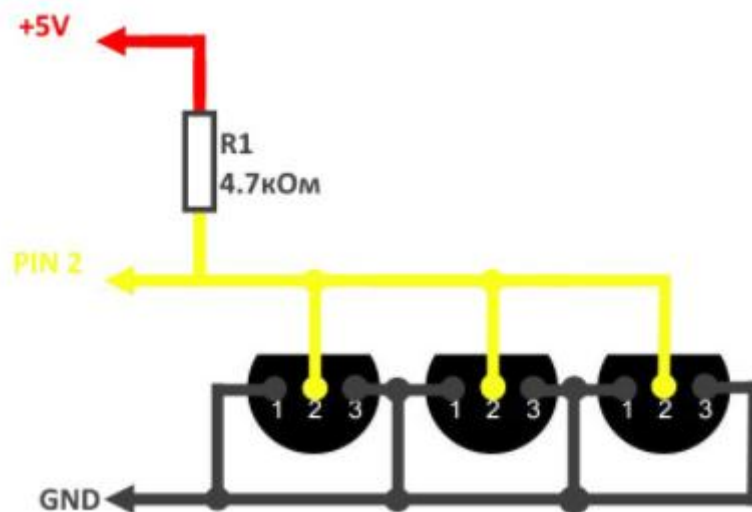


Рисунок 2.5 – Схема паразитного включення DS18B20

Для того, щоб зрозуміти принципи роботи датчика, необхідно ознайомитися з його внутрішньою складовою. На рис. 2.6 показана структурна схема датчика DS18B20.

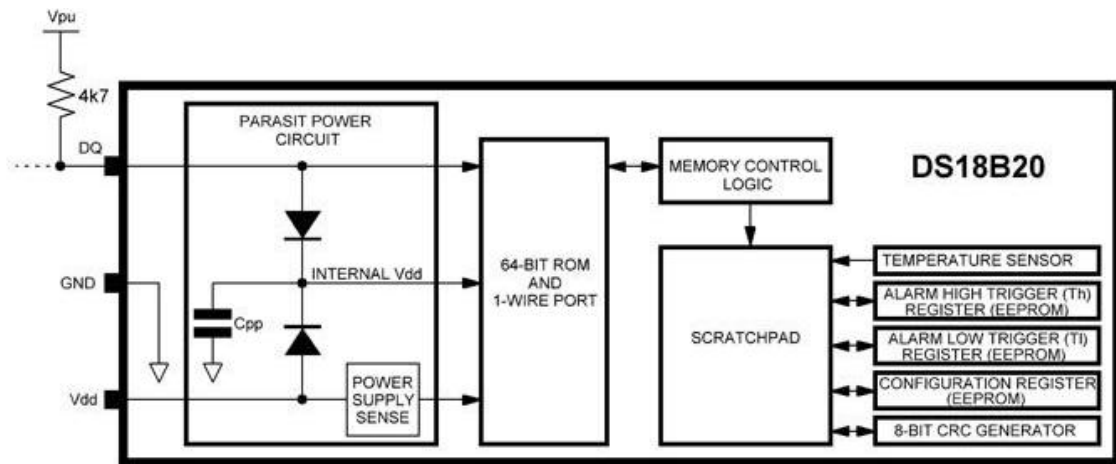


Рисунок 2.6 – Структурна схема датчика DS18B20

Як описується, вище ця мікросхема має два режими живлення: режим прямого живлення (напруга подається на виводи Vdd і GND) і режим паразитного живлення (датчик живиться від лінії даних).

Зі структурної схеми видно, що за пряме живлення відповідає блок (POWER SUPPLY SENSE), а режим паразитного живлення забезпечує блок (PARASIT POWER CIRCUIT) в складі якого основну роль грає конденсатор Cpp, як буфер живлення.

Далі слідує модуль "64-BIT ROM AND 1-WIRE PORT". У ньому міститься інформація про унікальну адресу кожного пристрою. Даний код записується заводом-виготовлювачем у внутрішню енергонезалежну пам'ять датчика (EEPROM) і ніколи не повторюється. В цьому ж місці розміщена підсистема взаємодії з 1-Wire інтерфейсом.

Блок "MEMORY CONTROL LOGIC" здійснює взаємозв'язок між командами інтерфейсу 1-Wire та внутрішньою пам'яттю датчика SCRATCHPAD. Дана пам'ять, в свою чергу, взаємодіє з декількома спеціалізованими регістрами DS18B20, а саме:

- TEMPERATURE SENSOR - дозволяє зчитувати перетворене значення температури.

- ALARM HIGH TRIGGER і ALARM LOW TRIGGER - ці регістри дозволяють виставити верхній і нижній поріг спрацьовування сигналів тривоги при виході температури за вказані межі.

- CONFIGURATON REGISTER - даний регістр призначений для настройки роздільної здатності температурного датчика. Регістр може бути налаштований на виконання перетворення температури з точністю 9 біт, 10 біт, 11 біт або 12 біт, що відповідає точності вимірювання 0.5°C, 0.25°C, 0.125°C і 0.0625°C [8].

- 8-BIT CRC GENERATOR - даний регістр призначений для генерації контрольної суми з метою підвищення захисту даних.

2.3 Обґрунтування вибору програмного забезпечення

2.3.1 Вибір програми візуального програмування контролерів

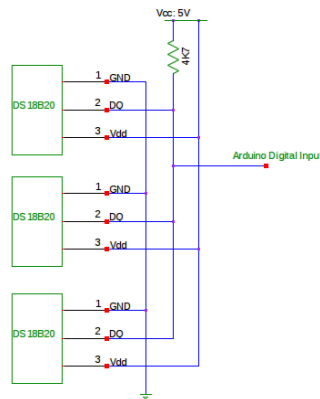
NodeMcu досить популярна платформа, це означає що існує велика кількість програмного забезпечення яке дозволить запрограмувати цей мікроконтролер. Кожне середовище розробки має свої переваги й недоліки, та різний функціонал і засоби роботи. При виборі ПЗ в даному випадку треба орієнтуватися на його складність та різноманітний функціонал, і під ці два критерії ідеально підходить програма FLProg.

FLProg - система візуального програмування контролерів. За допомогою цієї програми можна запрограмувати контролер не знаючи текстових мов програмування, схема генерується в середовищі проектування [7].

Ця програма повністю русифікована що дозволяє вільно користуватися довідкою у якій детально описані інструкції до використання того або іншого елементу чи блоку. Довідка зображена на рис 2.7

Блок датчика температуры DS18B20 (DS18S20, DS1822)

Блок позволяет использовать в проекте датчики температуры DS18B20, DS18S20, DS1822. Датчики подключаются шине OneWire согласно схеме.



В проекте может быть реализовано несколько шин OneWire. В диалоге настройки блока необходимо указать шину OneWire к которой подключен датчик, адрес датчика и тип датчика

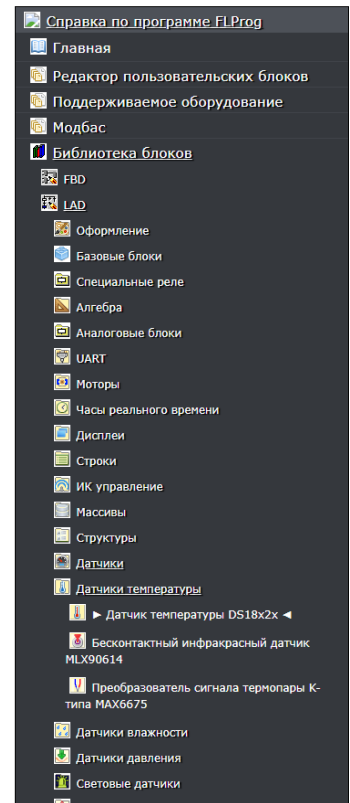
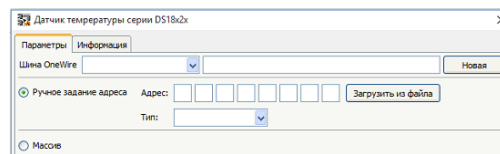


Рисунок 2.7 – Довідка програми FLProg

Візуальні мови програмування FBD і Ladder Diagram (LD, LAD, PKC), за допомогою яких пишеться програма, використовуються для програмування практично всіх логічних реле, і частини промислових контролерів у всьому світі [9, 10].

При створенні нового проекту вам запропонують вибрати мову програмування, на якій ви будете створювати проект, і контролер, на якому цей проект буде реалізований, вікно вибору зображено на рис 2.8. У нашому випадку вибираємо мову програмування FBD та Wi-Fi модуль ESP-8266 NodeMcu v3.

FBD (Function Block Diagram) - це стандартна мова графічного програмування. Програма складається з ланцюгового списку, який виконується послідовно зверху вниз. При програмуванні використовуються набори бібліотечних блоків. Блок (елемент) - це підпрограми, функції або функціональні блоки (I, АБО, НЕ, тригери, таймери, лічильники, аналогові блоки обробки сигналів, математичні операції тощо). Кожен окремий ланцюг

це вираз, який складається графічно з окремих елементів. Для того щоб утворити ланцюг потрібно до виходу блоку підключити наступний блок. У середині ланцюга блоки виготовляються строго в порядку їх з'єднання. Результати розрахунків ланцюга будуть записані у внутрішню змінну або подані на вихід контролера [9, 10].

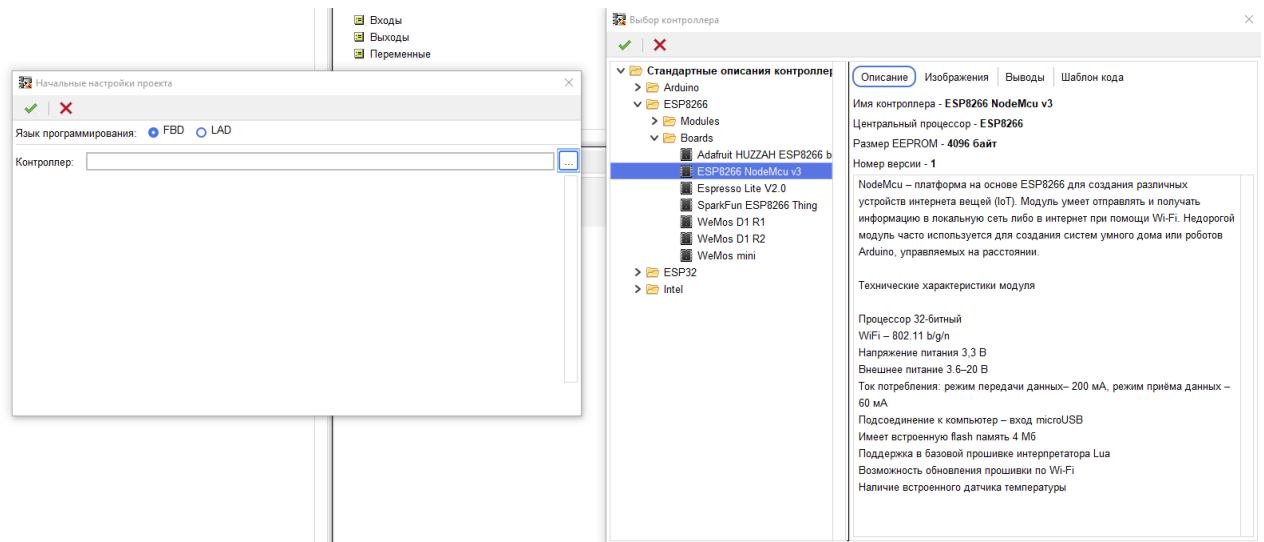


Рисунок 2.8 Вибір контролера та мови програмування

Проект в FLProg являє собою набір плат, на кожній з яких зібраний закінчений модуль загальної схеми. Для зручності роботи кожна плата має найменування і коментарі. Так само кожен модуль можна згорнути (для економії місця в робочій зоні, коли робота над нею закінчена) і розгорнути [9, 10].

У правій частині робочої зони розташована бібліотека елементів. У схему елементи переносяться простим перетягуванням. При подвійному натисканні по елементу буде показана інформація про нього (див. рисунок 2.9) [9, 10].

Ladder Diagram - мова релейних або сходових діаграм являє собою просту у зверненні, графічну мову розробки. В його основі лежать релейно-контактні схеми, тому елементами логіки тут виступають: обмотки реле, контакти реле, горизонтальні та вертикальні перемички. Ця мова орієнтована

на інженерів по автоматизації промислових компаній. Для логіки контролера передбачений чіткий інтерфейс, який не тільки полегшує завдання самопрограмування та налагодження, але й спрощує швидке усунення несправностей обладнання, підключеного до контролера. [9, 10, 11, с 115].

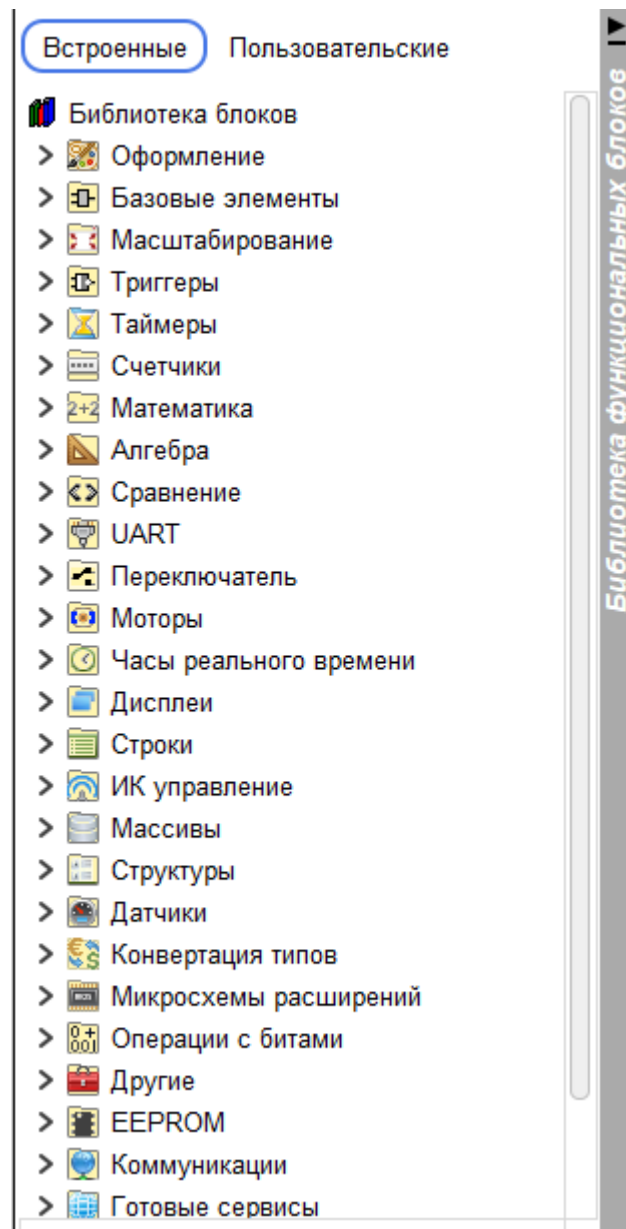


Рисунок 2.9 – Бібліотека блоків

Програма на мові релейної логіки має досить інтуїтивно зрозумілий інженерам-електрикам інтерфейс, який представляє логічні операції, наприклад як електричний ланцюг з замкнутими і роз'єднаними контактами. Перебіг або відсутність струму в цьому ланцюзі відповідає результату

логічної операції (істина – якщо струм тече; брехня – якщо струм не тече).

Пари контактів реле або кнопки – ось основні логічні змінні мови LD, при цьому стан змінних – це є ні що інше, як стан контактів: розімкнутий або замкнутий [9, 10, 11].

2.3.2 Вибір ПЗ для операційної системи ANDROID

Існує багато способів віддалено підключатися до приладів, щоб отримати з них дані необхідні дані, вимкнути або увімкнути певні виконавчі механізми. Дуже зручно використовувати для систем розумного будинку, для моніторингу стану якихось приладів. Весь цей комплект завдань вміє вирішувати HMI-KaScada на Android, яка дозволяє приєднатися до пристрою по одному з доступних способів зв'язку. Було прийнято рішення вибрати саме цю mini-SCADA тому що вона напряду працює з вже вибраним середовищем проектування FLProg, та процес їх з'єднання не складе труднощів. Програма досить універсальна, підтримує всі пристрої, які мають відношення до Modbus RTU. Дизайн досить сучасний, ліцензія цієї програми зовсім не дорога, абсолютно будь-якому користувачеві буде по кишені придбати їх та управляти пристроєм на відстані. Також HMI KaScada може працювати одночасно з 16 пристроями, підключеними по різним каналам зв'язку.

Підтримуються наступні види зв'язків:

- WiFi - MODBUS TCP, MODBUS RTU OVER TCP;
- Bluetooth - MODBUS RTU;
- USB - MODBUS RTU.

Розглянемо основні компоненти KaScada. Усі компоненти розбиті по групах за своїми основними функціями. На рис. 2.10. зображена перша сторінка елементів для прийняття, обробки та передачі даних.

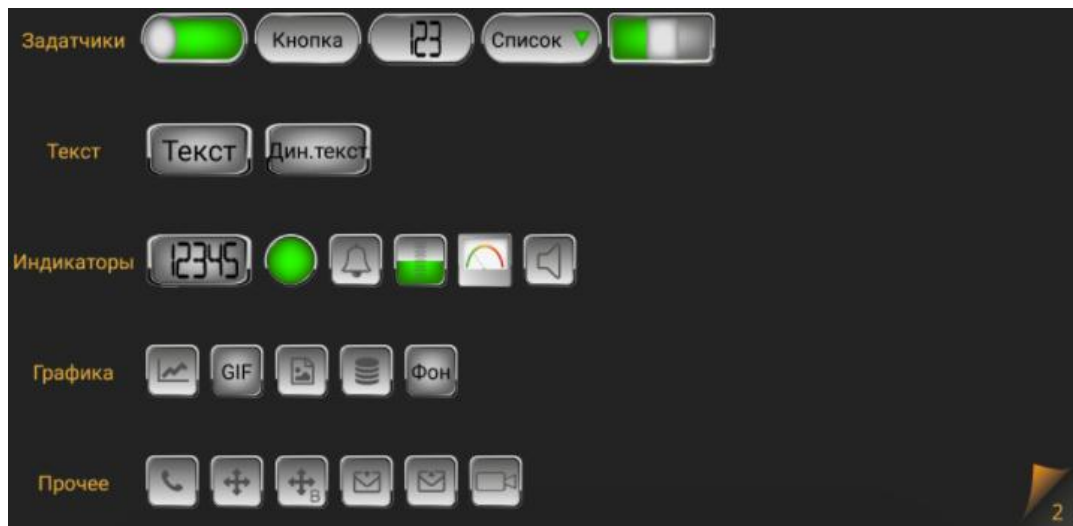


Рисунок 2.10 – Елементи для прийняття, обробки та передачі даних

На рис. 2.11 зображена друга сторінка функцій та функціональних блоків для створення елементарної логіки.

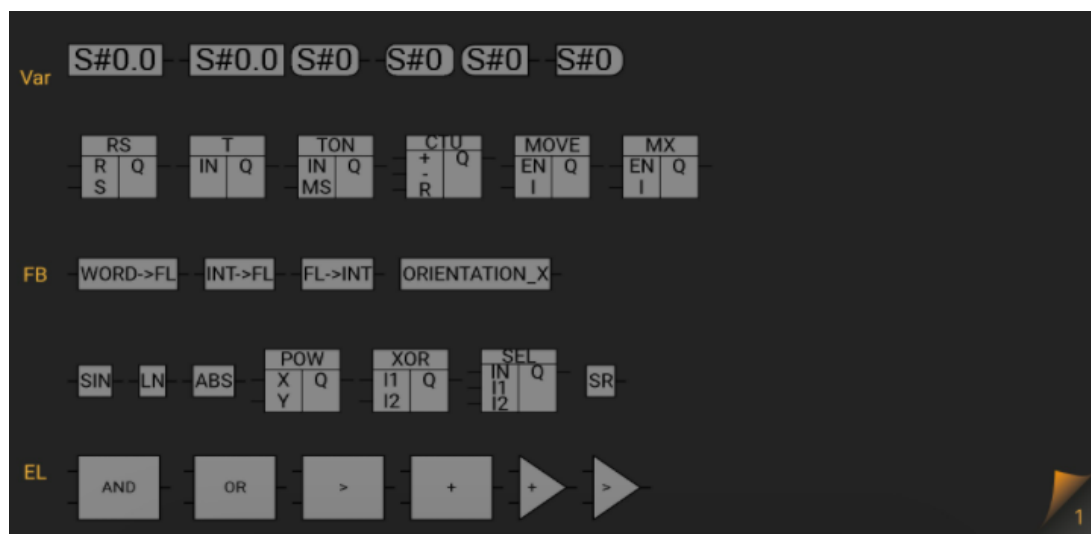


Рисунок 2.11 – Функції та функціональні блоки для створення елементарної логіки

Розглянемо основні можливості KaScada:

- отримання даних з GPS для визначення часу сходу і заходу сонця;
- наявність WEB-сервер для перегляду, управління і завантаження архівів;
- отримання даних з датчика нахилу по трьох осях;

- отримання даних з датчиків з MODBUS протоколом (датчики температури, тиску, вологості);
- наявність системних реєстрів;
- підключення по RTSP до IP-камерам;
- наявність функція, яка приймає і відправляє СМС (можна встановити уставку);
- наявність відтворення аудіо;
- можливість писати архів, виводити архів в графік, експортувати архів в csv форматі (Excel) і відправляти цей файл через електронну пошту;
- наявність відтворення GIF-анімації;
- наявність MODBUS Slave, можливе підключення та управління безлічі master-slave;
- у наявність хмарний шлюз, який розв'язувати проблеми з сірими та динамічними IP;
- є можливість написати власну логіку за допомогою вбудованого FBD (понад 20 блоків);

2.4 Проектування дистанційного терморегулятора

Для початку потрібно розробити схему терморегулятора і з'єднати його з Wi-Fi модулем ESP-8266 NodeMcu v3 (рисунок 2.12), Лістинг програми наведений у Додатку А , а експериментальна модель у додатку Б.

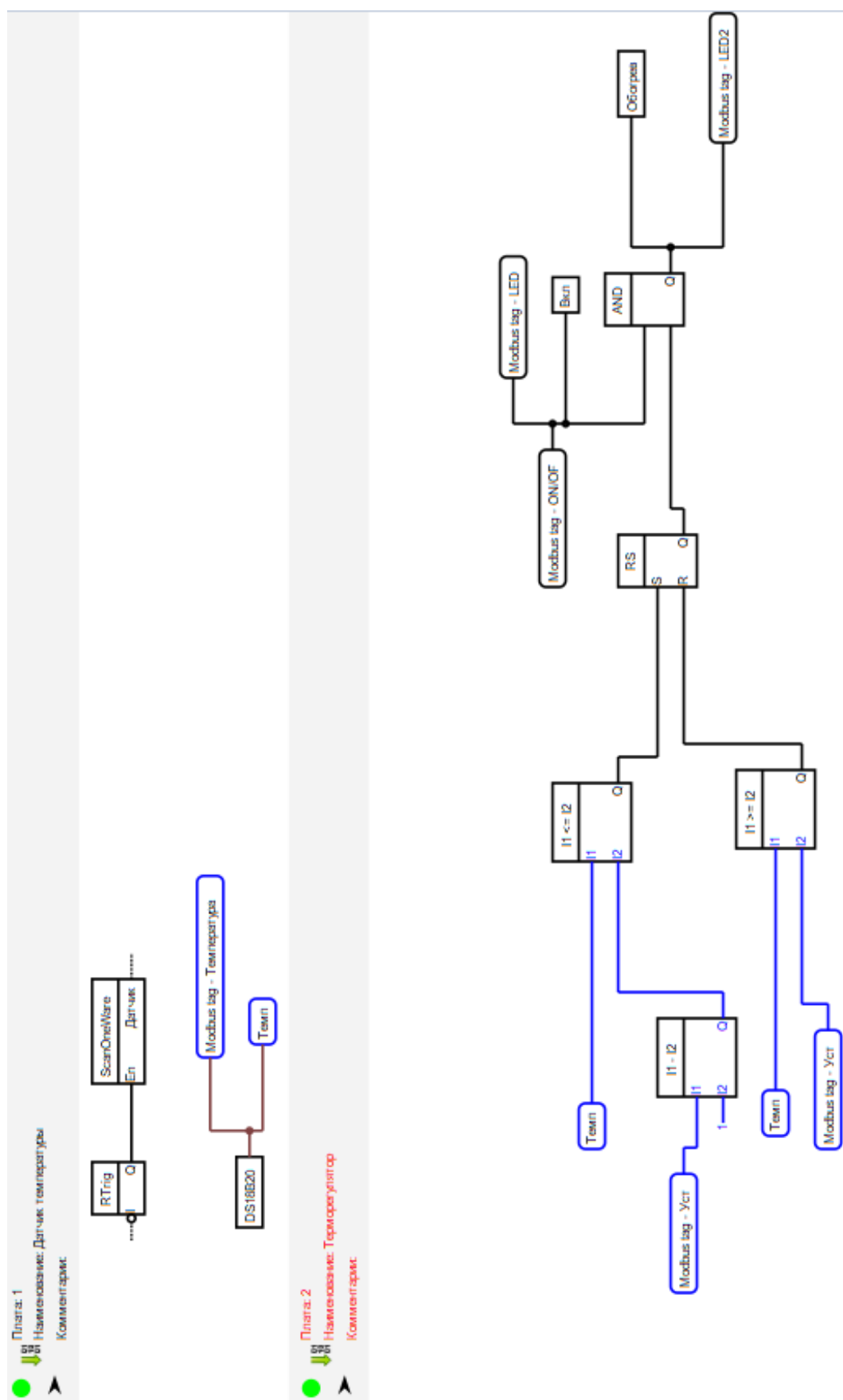


Рисунок 2.12 – Модель терморегулятора у FLProg

Далі з'єднуємо схему з сервером Kascada-Cloud (рисунок 2.13)

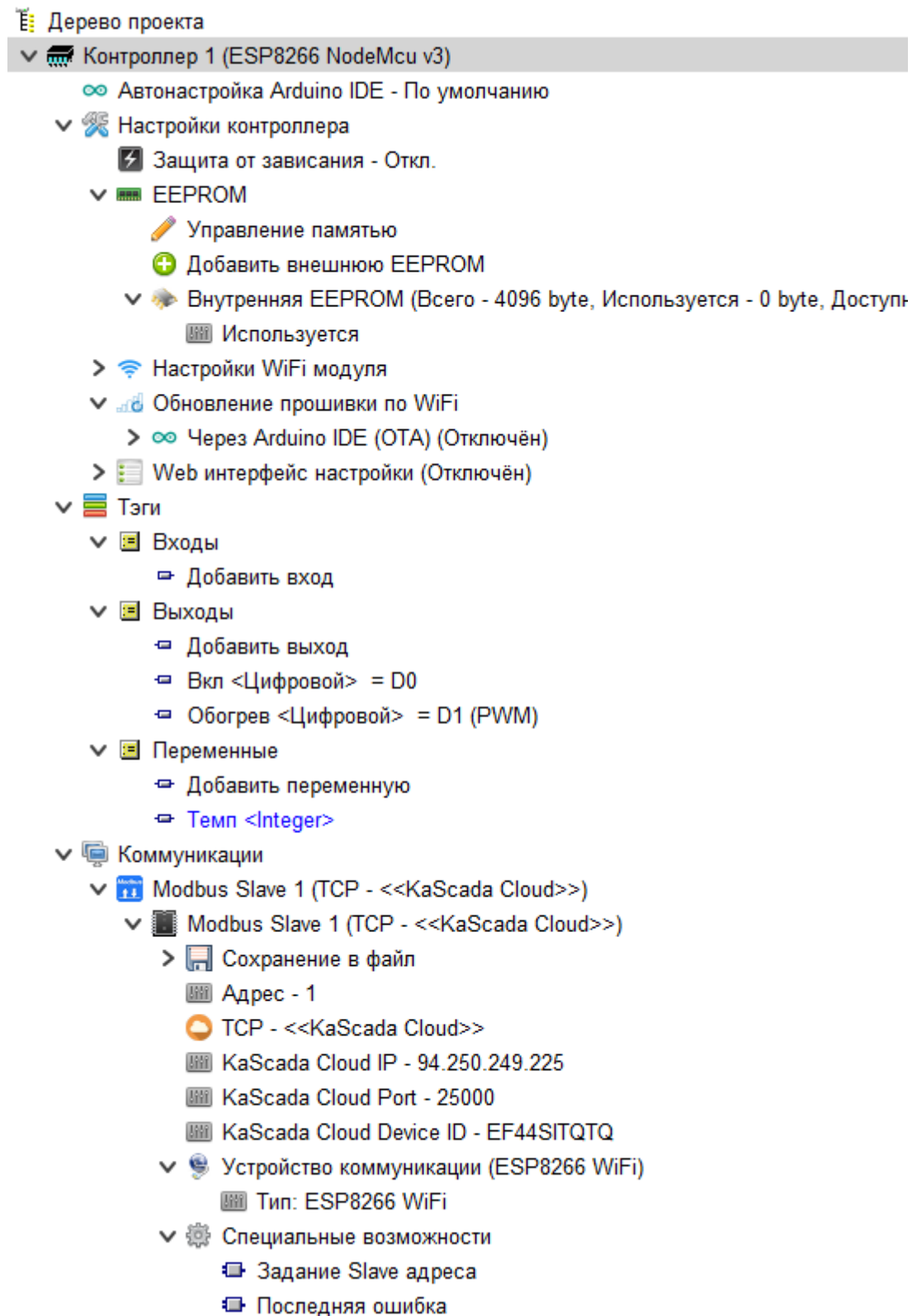


Рисунок 2.13 – Налаштування з'єднання схеми з сервером

Останній крок: створення схеми керування нагрівальним елементом у Mini-SCADA HMI-KaScada, для дистанційного керування обігрівом за допомогою смартфона з операційною системою ANDROID 5.1+ (рисунок 2.14).



Рисунок 2.14 – Інтерфейс дистанційного керування температурою

ВИСНОВКИ

Була розроблена система моніторингу та керування температури, яка реалізованою за допомогою програми FLProg з посиланням на Mini-SCADA HMI-KaScada, через Kascada-Cloud. Даний проєкт можна використовувати у SMART HOUSE, на прикладі цього проєкту можна дистанційно керувати будь яким об'єктом, від вентилів до вітрових та сонячних електростанцій, з системою планування та управління.

Проаналізовані всі види забезпечення проєктованих автоматизованих систем, а саме: математичне, лінгвістичне, технічне, інформаційне та програмне, крім того було розроблено методичне забезпечення для проєктування аналогічних систем.

Керування термостатом здійснюється у мобільній версії SCADA, оскільки мобільні пристрої з кожним роком стають все продуктивніше. Сьогодні працівники, які мають при собі смартфон або інший пристрій на базі ОС Android, можуть отримати інформацію і доступ до управління виробництвом практично з будь-якої точки світу. Програми дозволяють дистанційно здійснювати візуальний контроль над промисловими приладами (компресорами, витратомірами).

Покращення НМІ в таких областях, як людино-орієнтоване проєктування дозволяють інженерам - розробникам інтерфейсів і операторам ще більше підвищити продуктивність своєї праці. В основі розробки сучасних НМІ лежать дослідження, що проводяться в рамках згаданого вище підходу HCD. Вони включають колективну роботу вчених, розробників і постачальників техніки і автоматики, виробників технологічного обладнання, а також безпосередньо користувачів.

Робота пройшла апробацію на конференції Тиждень науки; проєкт реалізовано, розроблено дослідний зразок (див. додаток Б) та проведено необхідні експерименти для перевірки його функціоналу.

ПЕРЕЛІК ПОСИЛАНЬ

1. **Бобух А. О.** Автоматизовані системи керування технологічними процесами [Текст]: навч. посібник / А. О Бобух; ХНАМГ. – Харків: ХНАМГ, 2006. – 65с. - Режим доступу: https://eprints.kname.edu.ua/9106/1/АСК_ТП.pdf [ukr.]
2. **Паламар, Михайло.** Проектування комп'ютеризованих вимірювальних систем і комплексів. [Текст] / Михайло Паламар, Михайло Стрембіцький, Андрій Паламар; Тернопільський національний технічний університет імені Івана Пулюя, 2018. – 150 с.
3. Manufacturing Execution Systems. [Electronic resource]. - Access mode: https://books.google.com.ua/books?hl=ru&lr=&id=ilqVDw-zWcEC&oi=fnd&pg=PP15&dq=Manufacturing+Execution+Systems&ots=Larwvh2vaM&sig=s2Xj6YTIIdTP3zAFyv6Bag0Brno&redir_esc=y#v=onepage&q=Manufacturing%20Execution%20Systems&f=false
4. LIMS - [Електронний ресурс]. - Режим доступу: <https://www.autoscribeinformatics.com/resources/blog/what-is-a-lims>[eng.]
5. Розробка людино-машинних інтерфейсів та систем збору даних з використанням програмних засобів SCADA/HMI. [Електронний ресурс]. - Режим доступу: <https://drive.google.com/file/d/1r4MbMk0EALjGQsxgoCWCUqO4wXSDQf5f/view>[ukr.]
6. Введення в систему SCADA. [Електронний ресурс]. - Режим доступу: <https://uk.answersexpress.com/an-introduction-scada-systems-99185>[ukr.]
7. NodeMCU v3 ESP8266. [Електронний ресурс]. - Режим доступу: <https://myrobot.ru/wiki/index.php>[ukr.]
8. Датчик температури DS18B20. [Електронний ресурс]. - Режим доступу: <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>[eng.]

9. FBD [Електронний ресурс]. - Режим доступу: [https://flprog.ru/chto-takoe-flprog/\[rus.\]](https://flprog.ru/chto-takoe-flprog/[rus.])
10. Види середовищ програмування. [Електронний ресурс]. - Режим доступу: [https://foktver.ru/uk/internet/sreda-programmirovaniya-arduino-blokami-programmirovanie-arduino-s/\[ukr.\]](https://foktver.ru/uk/internet/sreda-programmirovaniya-arduino-blokami-programmirovanie-arduino-s/[ukr.])
11. **Петров И. В.** ПЗО Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / В. П. Дьяконова. – М.: СОЛОН-Пресс, 2004. – 256 с : ил. - (Серия «Библиотека инженера»)
12. **Елизаров, И. А.** Интегрированные системы проектирования и управления [Текст]: 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.
13. 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>
14. SCADA системи [Електронний ресурс]. - Режим доступу: [http://www.scadasystems.net/\[eng.\]](http://www.scadasystems.net/[eng.])
15. Як вибрати SCADA систему [Електронний ресурс]. - Режим доступу: [http://readonline.com.ua/items/47871-yak-vibratiscada-sistemu/\[ukr.\]](http://readonline.com.ua/items/47871-yak-vibratiscada-sistemu/[ukr.])
16. 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.
17. 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.

ДОДАТОК А

```

#include <ESP8266WiFi.h>
#include <OneWire.h>
extern "C"
{
    #include "user_interface.h"
}
bool _modbusSlaveDataTable_0[3];
int _modbusSlaveAddressTable_0[3] = {0, 1, 2};
int _modbusSlaveDataTable_4[2];
int _modbusSlaveAddressTable_4[2] = {1, 2};
byte _modbusSlaveBufferSize = 0;
byte _modbusSlaveBuffer[64];
const unsigned char _modbusSlave_fctsupported[] = {1, 5, 15, 3, 6, 16};
unsigned long _KaScadaCloudTimeOutStartTime;
IPAddress _KaScadaCloudIpAddress(94, 250, 249, 225);
WiFiClient _modbusSlaveTCPClient;
byte _modbusSlaveMBAPBuffer[6];
OneWire _ow4(4);
byte _FLPArray133289689[9];
bool ESPControllerWifiClient_HRD = 0;
bool ESPControllerWifiClient_status = 1;
bool ESPControllerWifiClient_isDHCP = 1;
bool ESPControllerWifiClient_IsNeedReconnect = 0;
bool ESPControllerWifiClient_workStatus = 1;
char ESPControllerWifiClient_SSID[40] = "";
char ESPControllerWifiClient_password[40] = "";
IPAddress ESPControllerWifiClient_ip(0, 0, 0, 0);
IPAddress ESPControllerWifiClient_dns(0, 0, 0, 1);
IPAddress ESPControllerWifiClient_gateway(0, 0, 0, 1);
IPAddress ESPControllerWifiClient_subnet(255, 255, 255, 0);
uint8_t ESPControllerWifiClient_mac[6] = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0};
int _gtv1;
bool _sowbl_needScan = 0;
bool _sowbl_ost = 0;
bool _sowbl_Out_1 = 0;
bool _trgrt1 = 0;
bool _trgrt1I = 0;
bool _trgs1 = 0;
unsigned long _d18x2x1Tti = 0UL;
float _d18x2x1O = 0.00;

```

```

bool _tempVariable_bool;
int _tempVariable_int;
float _tempVariable_float;
void setup()
{
    pinMode(16, OUTPUT);
    digitalWrite(16, 0);
    pinMode(5, OUTPUT);
    digitalWrite(5, 0);
    WiFi.mode(WIFI_STA);
    _esp8266WifiModuleClientReconnect();
    _KaScadaCloudTimeOutStartTime = millis() - 5000;
}
void loop()
{
    if (_isTimer(_KaScadaCloudTimeOutStartTime, 5000))
    {
        if(_modbusSlaveTCPClient.connect(_KaScadaCloudIpAddress, 25000))
        {
            _modbusSlaveTCPClient.print("EF44SITQTQ");
            _KaScadaCloudTimeOutStartTime = millis();
        }
    }
    _modbusSlavePoll();
    if(ESPControllerWifiClient_IsNeedReconnect)
    {
        _esp8266WifiModuleClientReconnect();
        ESPControllerWifiClient_IsNeedReconnect = 0;
    }
    ESPControllerWifiClient_status = WiFi.status() == WL_CONNECTED;
    if(ESPControllerWifiClient_status)
    {
        if(!ESPControllerWifiClient_HRD)
        {
            ESPControllerWifiClient_ip = WiFi.localIP();
            ESPControllerWifiClient_subnet = WiFi.subnetMask();
            ESPControllerWifiClient_gateway = WiFi.gatewayIP();
            ESPControllerWifiClient_dns = WiFi.dnsIP();
            WiFi.macAddress(ESPControllerWifiClient_mac);
            ESPControllerWifiClient_HRD = 1;
        }
    }
}

```

```

else
{
    ESPControllerWifiClient_HRD = 0;
}
//Плата:1
//Наименование:Датчик температуры
if(!(!0))
{
    if(!_trgrtl)
    {
        _trgrtl = 0;
    }
    else
    {
        _trgrtl = 1;
        _trgrtlI = 1;
    }
}
else
{
    _trgrtl = 0;
    _trgrtlI = 0;
}
;
if(_sowbl_needScan)
{
    if(_oneWireSearch(_FLPArray133289689, _ow4))
    {
        _sowbl_Out_1=1;
    }
    _ow4.reset_search();
    _sowbl_needScan=0;
}
if(_trgrtl)
{
    if(!_sowbl_ost)
    {
        _sowbl_ost=1;
        _sowbl_needScan=1;
        _sowbl_Out_1=0;
    }
}
}

```

```

else
{
    _sowbl_ost=0;
}
if(!_isTimer(_d18x2x1Tti, 1000))
{
    _d18x2x1Tti=millis();
    _tempVariable_float = _readDS18_ow4(_FLPArray133289689, _FLPArray133289689[8]);
    if (_tempVariable_float < 500)
    {
        _d18x2x1O = _tempVariable_float;
    }
}
_tempVariable_int = (_d18x2x1O);
_modbusSlaveDataTable_4[0] = _tempVariable_int;
_gtv1 = (_d18x2x1O);
//Плата:2
//Наименование:Терморегулятор
if((_gtv1)<=((_modbusSlaveDataTable_4[1])-(1))) _trgs1 = 1;
if((_gtv1)>=((_modbusSlaveDataTable_4[1]))) _trgs1 = 0;
digitalWrite(16, (_modbusSlaveDataTable_0[0]));
_tempVariable_bool = (_modbusSlaveDataTable_0[0]);
_modbusSlaveDataTable_0[1] = _tempVariable_bool;
digitalWrite(5, (((_modbusSlaveDataTable_0[0])) && (_trgs1)));
_tempVariable_bool = (((_modbusSlaveDataTable_0[0])) && (_trgs1));
_modbusSlaveDataTable_0[2] = _tempVariable_bool;
}
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));
    }
}
byte _modbusSlavePoll()
{

```

```

        return process_modbus_FC16();
        break;
    default:
        break;
    }
    return 25;
}

byte _modbusValidateRequest()
{
    boolean isSupported = false;
    for (uint8_t i = 0; i < sizeof(_modbusSlave_fctsupported); i++)
    {
        if (_modbusSlave_fctsupported[i] == _modbusSlaveBuffer[1])
        {
            isSupported = 1;
            break;
        }
    }
    if (!isSupported)
    {
        return 1;
    }
    int intRegs = 0;
    byte byteRegs;
    switch (_modbusSlaveBuffer[1])
    {
        case 1:
            if (!(checkModbusRange(word(_modbusSlaveBuffer[2], _modbusSlaveBuffer[3]),
word(_modbusSlaveBuffer[4], _modbusSlaveBuffer[5]), 0)))
            {
                return 2;
            }
            break;
        case 5:
            if (!(checkModbusAddres(word(_modbusSlaveBuffer[2], _modbusSlaveBuffer[3]), 0)))
            {
                return 2;
            }
            break;
        case 15:
            if (!(checkModbusRange(word(_modbusSlaveBuffer[2], _modbusSlaveBuffer[3]),
word(_modbusSlaveBuffer[4], _modbusSlaveBuffer[5]), 0)))

```

```

    {
        return 2;
    }
    break;
case 6 :
    if(!(checkModbusAddres((word(_modbusSlaveBuffer[2],_modbusSlaveBuffer[3])),4)))
    {
        return 2;
    }
    break;
case 3 :
case 16 :
    if(!(checkModbusRange((word(_modbusSlaveBuffer[2],_modbusSlaveBuffer[3])),
(word(_modbusSlaveBuffer[4],_modbusSlaveBuffer[5])),4)))
    {
        return 2;
    }
    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 0:
            tableSize = 3;
            break;
        case 4:
            tableSize = 2;
            break;
    }
    for (byte i = 0; i < tableSize; i++)
    {
        if((modbusSlaveAddresFromIndex(i,table)) == addr)
        {
            return i;
        }
    }
}

```

```

    }
}
return -1;
}
int modbusSlaveAdresFromIndex(byte index, byte table)
{
    switch (table)
    {
        case 0:
            return _modbusSlaveAdresTable_0[index];
            break;
        case 4:
            return _modbusSlaveAdresTable_4[index];
            break;
    }
    return -1;
}
bool checkModbusRange(int startAddr, int addrNumber, byte table)
{
    for (int i=0; i < addrNumber; i++)
    {
        if(!(checkModbusAdres((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;
    }
}

```

```

String stringBuffer = "";
_modbusSlaveMBAPBuffer[4] = highByte(_modbusSlaveBufferSize);
_modbusSlaveMBAPBuffer[5] = lowByte(_modbusSlaveBufferSize);
for (int i=0; i<6; i++)
{
    stringBuffer.concat(char(_modbusSlaveMBAPBuffer[i]));
}
for (int i=0; i<_modbusSlaveBufferSize; i++)
{
    stringBuffer.concat(char(_modbusSlaveBuffer[i]));
}
_modbusSlaveTCPClient.print(stringBuffer);
_modbusSlaveBufferSize = 0;
}
byte _modbusGetSlaveRxBuffer()
{
    byte currentByte = 0;
    boolean bBuffOverflow = false;
    byte currentByteIndex = 0;
    _modbusSlaveBufferSize = 0;
    while (_modbusSlaveTCPClient.available())
    {
        currentByte = _modbusSlaveTCPClient.read();
        if (currentByteIndex < 6)

        {
            _modbusSlaveMBAPBuffer[currentByteIndex] = currentByte;
        }
        else
        {
            _modbusSlaveBuffer[_modbusSlaveBufferSize] = currentByte;
            _modbusSlaveBufferSize++;
            if (_modbusSlaveBufferSize >= 64) bBuffOverflow = true;
        }
        currentByteIndex++;
    }
    if (bBuffOverflow)
    {
        return -3;
    }
    return _modbusSlaveBufferSize;
}

```

```

byte process_modbus_FC1(byte table)
{
    byte bytesNo, bitsNo;
    int currentCoil, coil;
    bool value;
    byte index;
    int startCoil = word(_modbusSlaveBuffer[2], _modbusSlaveBuffer[3]);
    int coilNo = word(_modbusSlaveBuffer[4], _modbusSlaveBuffer[5]);
    bytesNo = byte(coilNo / 8);
    if (coilNo % 8 != 0) bytesNo++;
    _modbusSlaveBuffer[2] = bytesNo;
    _modbusSlaveBufferSize = 3;
    bitsNo = 0;
    for (currentCoil = 0; currentCoil < coilNo; currentCoil++)
    {
        coil = startCoil + currentCoil;
        index = modbusSlaveIndexForAddress(coil, table);
        if (table == 0)
        {
            value = _modbusSlaveDataTable_0[index];
        }
        bitWrite(_modbusSlaveBuffer[_modbusSlaveBufferSize], bitsNo, value);
        bitsNo++;
        if (bitsNo > 7)
        {
            bitsNo = 0;
            _modbusSlaveBufferSize++;
        }
    }
    if (coilNo % 8 != 0) _modbusSlaveBufferSize++;
    _modbusSlaveSendTxBuffer();
    return _modbusSlaveBufferSize + 2;
}

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 = modbusSlaveIndexForAddress(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_FC5()
{
    byte index;
    int coil = word(_modbusSlaveBuffer[2], _modbusSlaveBuffer[3]);
    index = modbusSlaveIndexForAddress(coil, 0);
    _modbusSlaveDataTable_0[index] = (_modbusSlaveBuffer[4] == 0xff);
    _modbusSlaveBufferSize = 6;
    _modbusSlaveSendTxBuffer();
    return _modbusSlaveBufferSize + 2;
}

byte process_modbus_FC15()
{
    byte frameByte, bitsNo;
    byte currentCoil, coil;
    byte index;
    int startCoil = word(_modbusSlaveBuffer[2], _modbusSlaveBuffer[3]);
    int coilNo = word(_modbusSlaveBuffer[4], _modbusSlaveBuffer[5]);
    bitsNo = 0;
    frameByte = 7;
    for (currentCoil = 0; currentCoil < coilNo; currentCoil++)
    {
        coil = startCoil + currentCoil;
        index = modbusSlaveIndexForAddress(coil, 0);
        _modbusSlaveDataTable_0[index] = bitRead(_modbusSlaveBuffer[frameByte], bitsNo);
        bitsNo++;
    }
}

```

```

        if (bitsNo > 7)
        {
            bitsNo = 0;
            frameByte++;
        }
    }
    _modbusSlaveBufferSize = 6;
    _modbusSlaveSendTxBuffer();
    return _modbusSlaveBufferSize + 2;
}

byte process_modbus_FC6()
{
    int address = word(_modbusSlaveBuffer[2], _modbusSlaveBuffer[3]);
    int index;
    index = modbusSlaveIndexForAddress(address, 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 = modbusSlaveIndexForAddress((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_ow4(byte addr[8], byte type_s)
{
    byte data[12];
    byte i;
    _ow4.reset();
    _ow4.select(addr);
    _ow4.write(0xBE);
    for (i = 0; i < 9; i++)
    {
        data[i] = _ow4.read();
    }
    _ow4.reset();
    _ow4.select(addr);
    _ow4.write(0x44, 1);
    if (_ow4.crc8(data, 8) != data[8])
    {
        return 501;
    }
    return _convertDS18x2xData(type_s, data);
}

bool _oneWireSeach(byte array[], OneWire ow)
{
    byte temp[8];
    byte i;

```

```

if (!ow.search(temp))
{
    return false;
}
if (OneWire::crc8(temp, 7) != temp[7])
{
    return false;
}
switch (temp[0])
{
    case 0x10:
        array[8] = 1;
        break;
    case 0x28:
        array[8] = 0;
        break;
    case 0x22:
        array[8] = 0;
        break;
    default:
        return false;
}
for (i = 0; i < 8; i++)
{
    array[i] = temp[i];
}
return true;
}

uint hexStrToInt(String instring)
{
    byte len = instring.length();
    if (len == 0) return 0;
    int result = 0;
    for (byte i = 0; i < 8; i++) // только первые 8 цифр влезут в uint32
    {
        char ch = instring[i];
        if (ch == 0) break;
        result <<= 4;
        if (isdigit(ch))
            result = result | (ch - '0');
        else result = result | (ch - 'A' + 10);
    }
}

```

```

    }
    return result;
}

void _esp8266WifiModuleClientReconnect()
{
    if(_checkMacAddres(ESPControllerWifiClient_mac))
    {
        wifi_set_macaddr(0, const_cast<uint8*>(ESPControllerWifiClient_mac));
    }
    if(ESPControllerWifiClient_isDHCP)
    {
        WiFi.config(0U, 0U, 0U, 0U, 0U);
    }
    else
    {
        WiFi.config(ESPControllerWifiClient_ip, ESPControllerWifiClient_gateway,
ESPControllerWifiClient_subnet, ESPControllerWifiClient_dns, ESPControllerWifiClient_dns);
    }
    WiFi.begin(ESPControllerWifiClient_SSID, ESPControllerWifiClient_password);
}

bool _checkMacAddres(byte array[])
{
    bool result = 0;
    for (byte i = 0; i < 6; i++)
    {
        if (array[i] == 255)
        {
            return 0;
        }
        if (array[i] > 0)
        {
            result = 1;
        }
    }
    return result;
}

void _parseMacAddressString(String value, bytearray[])
{
    int index;
    byte buffer[6] = {255, 255, 255, 255, 255, 255};
    byte raz = 0;

```

```

String tempString;
while ((value.length() > 0) && (raz <= 6))
{
    index = value.indexOf(":");
    if (index == -1)
    {
        tempString = value;
        value = "";
    }
    else
    {
        tempString = value.substring(0, index);
        value = value.substring(index + 1);
    }
    buffer[raz] = byte(hexStrToInt(tempString));
    raz++;
}
if (_checkMacAdres(buffer))
{
    for (byte i = 0; i < 6; i++)
    {
        array[i] = buffer[i];
    }
}
bool _compareMacAdresses(byte array1[], byte array2[])
{
    for (byte i = 0; i < 6; i++)
    {
        if (array1[i] != array2[i])
        {
            return 0;
        }
    }
    return 1;
}
bool _compareMacAdresWithString(byte array[], String value)
{
    byte buffer[6] = {255, 255, 255, 255, 255, 255};
    _parseMacAdresString(value, buffer);
    return _compareMacAdresses(array, buffer);
}

```

```

}
bool _checkMacAddressString(String value)
{
    byte buffer[6] = {255, 255, 255, 255, 255, 255};
    _parseMacAddressString(value, buffer);
    return _checkMacAddress(buffer);
}
String _macAddressToString(byte array[])
{
    String result = "";
    String temp = "";
    for (byte i = 0; i < 6; i++)
    {
        temp = String(array[i], HEX);
        if (temp.length() < 2)
        {
            temp = String("0") + temp;
        }
        result = result + temp;
        if (i < 5)
        {
            result = result + String(":");
        }
    }
    result.toUpperCase();
    return result;
}

```

ДОДАТОК Б

Загальний вигляд дослідного зразку

