МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій
(повне найменування інституту, назва факультету)

Кафедра комп'ютерної інженерії та електромеханіки
(повна назва кафедри)

ДИПЛОМНА БАКАЛАВРСЬКА РОБОТА

на тему

СИСТЕМА ВІДДАЛЕНОГО КОНТРОЛЮ ТЕМПЕРАТУРИ У ПРИМІЩЕННІ

Виконав: студент групи БКІ-19

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

Юрковський С.С
(прізвище та ініціали)

Керівник д.т.н., доц. Стаценко В.В.

(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій
Кафедра комп'ютерної інженерії та електромеханіки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Електромеханіка»

ЗАТВЕРДЖУЮ

Завідувач кафедри КІЕМ

_____проф. Злотенко Б.М.

"_____" _____2023 року

З А В Д А Н Н Я

НА ДИПЛОМНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Юрковському Сергію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема дипломної бакалаврської роботи **Система віддаленого контролю температури у приміщенні**

Науковий керівник роботи Стаценко Володимир Володимирович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
д.т.н., доцент

затверджені наказом вищого навчального закладу від 08.11.2022 № 224-уч.

2. Строк подання студентом роботи 1 червня 2023 року

3. Вихідні дані до дипломної бакалаврської роботи: Система віддаленого контролю температури у приміщенні.

4. Зміст дипломної бакалаврської роботи (перелік питань, які потрібно розробити): 1. Огля системи віддаленого контролю температури у приміщенні. 2. Описати конструкцію системи віддаленого контролю температури та підбір складових. 3.Описати роботу приладу.

5. Дата видачі завдання 01.02.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної бакалаврської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	10.02.2023	
2	Розділ 1. Огляд системи віддаленого контролю температури. датчики вимірювання температури	03.03.2023	
3	Розділ 2. Конструкція системи віддаленого контролю температури та підбір складових.	07.04.2023	
4	Розділ 3. Опис роботи приладу	12.05.2023	
6	Висновки	19.05.2023	
7	Оформлення дипломної бакалаврської роботи (чистовий варіант)	26.05.2023	
8	Здача дипломної бакалаврської роботи на кафедру для рецензування (за 14 днів до захисту)	29.05.2023	
9	Перевірка кваліфікаційної роботи на наявність текстових співпадінь та помилок (за 10 днів до захисту)	02.06.2023	
10	Подання дипломної бакалаврської роботи на затвердження завідувачу кафедри (за 7 днів до захисту)	06.06.2023	

Студент _____ Юрковський С.С.
(підпис) (прізвище та ініціали)

Науковий керівник роботи _____ Стаценко В.В
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Юрковський С. С. Система віддаленого контролю температури у приміщенні. – Рукопис.

Дипломна робота бакалавра за спеціальністю 123 – Комп'ютерна інженерія освітньої програми «Комп'ютерні системи та мережі». – Київський національний університет технологій та дизайну, Київ, 2023 рік.

Дана дипломна робота присвячена розробці системи віддаленого контролю температури у приміщенні з метою забезпечення комфортних умов для перебування людей та оптимізації споживання енергії в будівлях. У роботі проведено аналіз існуючих систем контролю температури та визначено їх переваги та недоліки.

На основі отриманих даних розроблено систему віддаленого контролю температури з використанням мікроконтролера та сенсорів температури. Результати експериментів показали ефективність розробленої системи та її можливості у плані забезпечення стабільної температури в приміщенні та зниження споживання електроенергії.

Отримані результати можуть бути корисними для компаній, які займаються розробкою та виробництвом систем віддаленого контролю за температурою, та дослідницьких груп, що працюють у сфері енергетичної ефективності та розробки інноваційних рішень для будівельної галузі.

Ключові слова: система віддаленого контролю, температура, мікроконтролер, сенсор, енергоефективність.

ABSTRACT

Yurkovskyi S. S. Remote temperature control system in the room - Manuscript.

Bachelor's thesis in the specialty 123 - Computer Engineering of the educational program "Computer Systems and Networks." - Kyiv National University of Technology and Design, Kyiv, 2023.

This thesis is devoted to the development of a system for remote control of indoor temperature in order to provide comfortable conditions for people and optimize energy consumption in buildings. The work analyzes existing temperature control systems and identifies their advantages and disadvantages.

Based on the data obtained, a remote temperature control system using a microcontroller and temperature sensors was developed. Experimental results have shown the effectiveness of the developed system and its capabilities in terms of ensuring a stable room temperature and reducing energy consumption.

The results obtained can be useful for companies engaged in the development and production of remote temperature control systems and research groups working in the field of energy efficiency and the development of innovative solutions for the construction industry

Keywords: remote control system, temperature, microcontroller, sensor, energy efficiency.

ЗМІСТ

ВСТУП

Актуальність роботи полягає у тому, що контроль за температурою є важливим фактором для забезпечення комфорту та здоров'я людей у приміщеннях. Через швидке зростання населення та збільшення обсягів будівництва приміщень, питання ефективного контролю за температурою стає все більш актуальним. Незабезпечення оптимальних температурних умов може призвести до погіршення здоров'я, підвищення рівня стресу та дискомфорту у людей.

Застосування систем віддаленого контролю за температурою дозволяє забезпечити точний контроль за температурою у віддалених приміщеннях, знизити витрати на енергоспоживання та отримати позитивний вплив на бізнес та приватний сектор. Тому дана дипломна робота має велике практичне значення для розробки оптимальної системи контролю за температурою у приміщеннях.

Метою роботи є розробка системи віддаленого контролю за температурою у приміщенні, яка буде забезпечувати точний контроль за температурним режимом, оптимізувати витрати на енергоспоживання та

підвищувати рівень комфорту та ефективності у віддалених приміщеннях. Для досягнення цієї мети у роботі поставлені наступні завдання: вивчити теоретичні основи систем контролю за температурою, проаналізувати існуючі рішення в цій області, розробити архітектуру системи та програмне забезпечення, реалізувати та протестувати систему на відповідність вимогам технічного завдання.

Об'єкт дослідження – система віддаленого контролю за температурою у приміщенні.

Предмет дослідження – розробка системи віддаленого контролю за температурою у приміщенні з використанням сучасних технологій програмування та IoT (Internet of Things) підходів.

Методи досліджень включають в себе аналіз наукової та технічної літератури, експериментальні дослідження, розробка програмного забезпечення, тестування та валідацію, аналіз результатів.

Інформаційною базою досліджень є навчальна та методична література, державні стандарти, а також відкриті джерела Internet.

Практичне значення отриманих результатів. У роботі запропоновано технічне рішення для створення та вдосконалення системи віддаленого контролю температури в приміщенні.

Апробація отриманих результатів полягає в створенні функціональної системи віддаленого контролю температури у приміщенні, яка дозволить надалі створювати умови для перебування людей та оптимізувати споживання енергії в будівлях.

Розроблена система може бути застосована у житлових, офісних, промислових та інших типах будівель, де контроль за температурою є важливим елементом енергетичної ефективності.

Структура та обсяг роботи. Дипломна робота бакалавра складається зі вступу, 3 розділів та висновків по них, загальних висновків, списку використаних джерел та додатків.

РОЗДІЛ 1. ОГЛЯД СИСТЕМИ ВІДДАЛЕНОГО КОНТРОЛЮ ТЕМПЕРАТУРИ. ДАТЧИКИ ВИМІРЮВАННЯ ТЕМПЕРАТУРИ.

1.1 Поняття «системи віддаленого контролю температури». Переваги та недоліки.

"Система дистанційного вимірювання температури" має складну структуру, яка дозволяє людям контролювати температуру в різних місцях за допомогою спеціалізованих температурних датчиків [3].

Зібрані дані можуть бути надіслані на віддалені сервери або хмарні сервіси для детального вивчення та контролю.

Використання системи віддаленого вимірювання температури в промисловості:

- Виробничі підприємства можуть використовувати систему віддаленого контролю температури для нагляду за режимами температури в приміщеннях, процесах охолодження та нагріву, а також для моніторингу обладнання, яке працює при певних температурних режимах.
- Використання системи може забезпечити вчасне виявлення перегрівання або зниження температури, що може спричинити несправності в обладнанні або негативно вплинути на якість виробництва.

Використання системи віддаленого вимірювання температури в сільському господарстві:

- У сільському господарстві система може бути використана для контролю температурних умов в оранжереях, теплицях або інших приміщеннях, де знаходяться рослини.

- Віддалений моніторинг температури дозволяє вчасно реагувати на зміни і підтримувати оптимальний температурний режим для зростання рослин, що сприяє врожайності та якості сільськогосподарських культур.

Використання системи віддаленого вимірювання температури в медицині:

- В лікарнях та медичних закладах система може використовуватися для контролю температури у приміщеннях, лабораторіях, камерах зберігання лікарських препаратів, а також для моніторингу пацієнтів.
- Віддалений контроль допомагає забезпечити безпеку та стабільність умов зберігання ліків та медичних матеріалів, а також допомагає у вчасному виявленні небезпечних змін температури у приміщеннях або на лікарняному ліжку пацієнта.

Таким чином, система віддаленого вимірювання температури має широкий спектр застосування в різних галузях і може бути використана для забезпечення оптимального температурного режиму в середовищі, забезпечення безпеки та ефективного використання ресурсів [4].

Рис. 1.1. Система дистанційного контролю температури і вологості через GSM

Впровадження системи дистанційного моніторингу температури дозволяє отримати наступні переваги:

1) Миттєвий моніторинг: Система надає вам інформацію про температуру в приміщенні або на об'єкті в режимі реального часу без необхідності бути присутнім на місці.

2) Легкість доступу та зручність: Дані про температуру можна зручно отримати за допомогою веб-інтерфейсу або мобільного додатку, доступного з будь-якого місця, де є підключення до мережі Інтернет, що полегшує ефективне управління та моніторинг.

Рис. 1.2. Інтелектуальний домашній бездротовий датчик температури

3) Спектр застосувань системи охоплює моніторинг температури в різних сферах, таких як житлові райони, офісні приміщення, магазини та медичні заклади.

4) Завдяки інтелектуальним налаштуванням і програмній логіці, система може автоматизувати операції з нагрівання або охолодження приладів, забезпечуючи при цьому оптимальну температуру навколишнього середовища.

Крім перерахованих переваг і застосувань системи віддаленого вимірювання температури, можна додати такі аспекти:

- Автоматизація та інтеграція: Система віддаленого контролю температури може бути інтегрована з іншими системами автоматизації, такими як системи освітлення, вентиляції, а також системи безпеки, що дозволяє створити комплексне рішення для керування всіма аспектами приміщення.

- Моніторинг енергоефективності: Віддалений контроль температури також дозволяє вести моніторинг енергоефективності системи опалення та охолодження. Аналіз отриманих даних може виявити проблеми зі споживанням енергії та допомогти вдосконалити роботу системи для економії енергоресурсів.

- Оповіщення та аварійні ситуації: Система віддаленого контролю температури може бути налаштована для автоматичного оповіщення про небажані зміни температури або виникнення аварійних ситуацій. Це дозволяє оперативно реагувати на проблеми та забезпечувати безпеку у приміщенні.

- Збір та аналіз даних: Система віддаленого контролю температури може забезпечити збір, зберігання та аналіз даних про температурний режим протягом тривалого періоду часу. Це дозволяє виявити тенденції, паттерни та залежності, що можуть бути використані для удосконалення системи та прийняття ефективних управлінських рішень.

- Складніші алгоритми керування: За допомогою системи віддаленого контролю температури можна реалізувати більш складні алгоритми керування, такі як прогнозування температури, регулювання в залежності від зовнішніх умов, оптимальне розподілення енергії та багато інших.

Ці аспекти доповнюють переваги і застосування системи віддаленого вимірювання температури, розширюючи її функціональні можливості та використання.

Але також «Система віддаленого контролю температури» не ідеальна і має свій ряд недоліків, ось деякі із них:

1) Системи дистанційного вимірювання температури можуть стикатися з обмеженнями через необхідність постійного підключення до Інтернету. Доступ до системи та передача даних вимагає надійного онлайн-з'єднання. Отже, перебої або відсутність доступу до інтернету можуть створювати перешкоди.
2) Передача даних про температуру по мережі може ставити питання щодо конфіденційності та безпеки даних, тому необхідні заходи захисту для запобігання несанкціонованому доступу до цієї інформації.
3) Впровадження та підтримка системи віддаленого вимірювання температури можуть вимагати витрат на обладнання, програмне забезпечення та інші технічні аспекти.

Загально кажучи, системи віддаленого вимірювання температури є потужним інструментом для контролю та моніторингу температурного стану об'єктів, приміщень та систем, проте, перед їх використанням, слід ретельно вивчити переваги та недоліки, а також врахувати особливості конкретного застосування та вимоги системи.

1.2 Типи датчиків вимірювання температури та їх особливості

Існує ряд різних типів датчиків вимірювання температури, які можуть бути використані в різних системах віддаленого вимірювання температури. Основні типи датчиків включають:

1) Термопари [2]: Термопари базуються на явищі термоелектричного ефекту, де виникає напруга при з'єднанні двох різних металів у вимірюваній точці. Зміна температури впливає на величину цієї напруги, що дозволяє виміряти температуру. Вони відзначаються високою точністю, широким діапазоном вимірювання та довговічністю.

Рис. 1.3. Стандартна термопара

2) Термістори: Термістори засновані на залежності опору від температури. Їх опір змінюється залежно від зміни температури, дозволяючи виміряти її значення. Існують два типи термісторів: з позитивним температурним коефіцієнтом (PTC), де опір збільшується зі зростанням температури, та з негативним температурним коефіцієнтом (NTC), де опір зменшується зі зростанням температури. Термістори характеризуються високою чутливістю та швидким відгуком на зміни температури.

Рис. 1.4. Термістор NTC 60R 1A D=11mm (MF72-60D11)

3) RTD (Resistance Temperature Detector): RTD використовують платинові опори, чия опір змінюється зі зміною температури. Вони характеризуються високою точністю, стабільністю та довговічністю. Зміна опору відбувається лінійно зі зміною температури, що спрощує їх калібрування та використання.

Рис. 1.5. RTDs, Resistance Temperature Detectors, RTD Manufacturer

4) Датчики на основі напівпровідників [1]: Датчики, такі як датчики типу DS18B20 [5], використовують напівпровідникові елементи для вимірювання температури. Вони володіють високою точністю, широким діапазоном вимірювання та цифровим виводом даних, що

спрощує їх інтеграцію з мікроконтролерами. Вони також характеризуються швидким відгуком на зміни температури та мають вбудовану адресацію для підключення декількох датчиків до одного шини [29].

Рис. 1.6. Датчик температури DS18B20 (18B20) цифровий

Кожен тип датчика має свої особливості, вимоги до живлення, діапазон вимірювання та точність. Вибір певного типу датчика залежить від конкретних вимог системи та особливостей застосування.

1.2.1 Вимоги до датчиків температури

Вимоги до датчиків температури в системі дистанційного контролю температури є важливим аспектом при їх виборі. Нижче наведено деякі загальні вимоги, які слід враховувати при виборі датчиків температури: Точність вимірювання: Однією з найважливіших вимог є точність температурного датчика. [6] Точність вимірювання впливає на надійність отриманих даних та їх відповідність фактичному значенню температури. Висока точність необхідна, особливо в місцях, де невелике відхилення від бажаної температури може мати серйозні наслідки.

Діапазон вимірювання: Температурні датчики мають широкий діапазон вимірювання, що дозволяє використовувати їх в широкому діапазоні температурних умов. Залежно від застосування, можуть знадобитися датчики з низьким або високим температурним діапазоном.

Стійкість до екстремальних умов: Датчики температури повинні бути стійкими до екстремальних умов, таких як висока або низька температура, вологість, вібрація або хімічний вплив. Вони повинні забезпечувати надійну роботу навіть в умовах, коли інші елементи можуть втратити ефективність.

Чутливість і швидкість реакції: Датчики температури повинні мати достатню чутливість, щоб виявляти навіть невеликі зміни температури. Крім того, важлива швидкість реакції датчика - його здатність швидко реагувати на зміну температури і передавати відповідні дані.

Вартість і доступність: Вибір датчика температури також залежить від фінансових обмежень і доступності на ринку. Важливо збалансувати вимоги та вартість, а також переконатися, що вибрані датчики легко доступні для закупівлі та підтримки.

Враховуючи ці вимоги, можна зробити обґрунтований вибір датчиків температури, які найкраще відповідають конкретним потребам системи дистанційного моніторингу температури.

1.3 Методи вимірювання температури

У цьому пункті ми розглянемо різні методи вимірювання температури, які використовуються в датчиках та системах контролю температури [7].

1) Терморезистивний метод

Терморезистивний метод базується на вимірюванні зміни електричного опору при зміні температури. Датчики, які використовують цей метод, називаються терморезисторами. Терморезистори змінюють свій опір відповідно до зміни температури, що дозволяє виміряти температуру за

допомогою зчитування опору терморезистора. Залежно від типу матеріалу, використаного в терморезисторі, можуть бути застосовані різні математичні формули для перетворення зчитаного опору на температуру.

2) Термопарний метод

Термопарний метод вимірювання температури базується на залежності між температурою і електродиференційною силою, яка виникає між двома контактними провідниками з різних матеріалів при їх нагріванні. Датчики температури на основі термопар вимірюють електродиференційну силу і перетворюють її у відповідне значення температури. Термопари мають високу точність, широкий діапазон вимірювання і використовуються в широкому спектрі застосувань.

3) Інфрачервоні методи

Інфрачервоні методи вимірювання температури використовують здатність об'єктів випромінювати інфрачервоне випромінювання, яке залежить від їх температури. Датчики інфрачервоного випромінювання реєструють це випромінювання і перетворюють його у відповідне значення температури. Інфрачервоні методи мають великий діапазон вимірювання і можуть використовуватись для вимірювання температур об'єктів у великій відстані.

4) Діелектричні методи

Діелектричні методи вимірювання температури використовують зміни діелектричних властивостей речовин при зміні температури. Датчики на основі діелектричних матеріалів здатні вимірювати температуру шляхом вимірювання зміни ємності, діелектричної проникності або інших параметрів діелектрика. Діелектричні методи вимірювання можуть бути використані для вимірювання температури у різних середовищах, включаючи рідини, гази та тверді речовини.

Ці різні методи вимірювання температури мають свої переваги та обмеження і можуть бути використані в залежності від конкретних потреб проекту та умов експлуатації. Вибір правильного методу залежить від точності, діапазону вимірювання, швидкості відгуку, витрат енергії та інших факторів, які варто враховувати при розробці системи віддаленого контролю температури.

1.3.1 Переваги та недоліки методів вимірювання температури

1. Терморезистивний метод:

Переваги:

- Висока точність та стабільність вимірювання.
- Можливість працювати в широкому діапазоні температур.
- Можливість компенсувати вплив зовнішніх факторів, таких як джерела жару.
- Незалежність від джерела живлення.

Недоліки:

- Потреба в компенсації зварювальних сполучень або інших ефектів.
- Обмежений швидкість відгуку.
- Обмежений діапазон вимірювання порівняно з деякими іншими методами.

2. Термопарний метод:

Переваги:

- Широкий діапазон вимірювання, включаючи високі температури.
- Висока швидкість відгуку.
- Міцність та стійкість до зовнішніх впливів.
- Можливість використання у важкодоступних місцях.

Недоліки:

- Потреба в компенсації зварювальних сполучень або інших ефектів.
- Недостатня точність порівняно з деякими іншими методами.
- Залежність від типу матеріалу та комбінації термопари.

3. Інфрачервоні методи:

Переваги:

- Можливість вимірювання температури у великій відстані.
- Швидкість вимірювання та відсутність контакту з об'єктом.
- Можливість вимірювання температур великого діапазону.

Недоліки:

- Вплив на точність вимірювання зовнішніх факторів, таких як вплив оточуючого освітлення.
- Обмеження вимірювання відображаючих або прозорих поверхонь.
- Обмеження вимірювання непрямих або недоступних областей.

4. Діелектричні методи:

Переваги:

- Можливість вимірювання температури у різних середовищах, включаючи рідини, гази та тверді речовини.
- Можливість вимірювання зміни ємності або діелектричної проникності для визначення температури.
- Висока стабільність та низький вплив зовнішніх факторів.

Недоліки:

- Залежність від діелектричних властивостей матеріалу, що може змінюватися з температурою.
- Обмежений діапазон вимірювання порівняно з іншими методами.
- Потреба в калібруванні та компенсації впливу оточуючої середи.

Це лише загальні переваги та недоліки кожного методу, і вони можуть варіюватись залежно від конкретних моделей та виробників датчиків. Важливо враховувати ці фактори при виборі найбільш підходящого методу для вашої системи віддаленого контролю температури [8].

1.3.2 Застосування методів вимірювання температури

Застосування різних методів вимірювання температури може бути різноманітним і залежить від конкретних потреб та вимог проекту. Ось деякі приклади застосування різних методів[9]:

1. Терморезистивний метод:
   - Медицина: Вимірювання температури тіла пацієнтів у медичних пристроях.
   - Промисловість: Моніторинг температури у виробничих процесах, де потрібна висока точність.
   - Дослідження: Вимірювання температури у наукових дослідженнях, експериментах та лабораторних умовах.
2. Термопарний метод:
   - Енергетика: Вимірювання температури в різних частинах енергетичних систем, наприклад, у турбінах та котлах.
   - Автомобільна промисловість: Моніторинг температури двигуна, вихлопних газів, охолоджуючої рідини тощо.
   - Виробництво: Вимірювання температури під час обробки металів, паяння або зварювання.
3. Інфрачервоні методи:
   - Безпека: Вимірювання температури людей в аеропортах, вхідних точках або місцях, де необхідно виявлення підвищеної температури.
   - Будівництво: Вимірювання температури будівельних матеріалів, систем опалення або охолодження.
   - Агропромисловість: Моніторинг температури у сільськогосподарських культурах, складах зберігання, теплицях тощо.
4. Діелектричні методи:
   - Медицина: Вимірювання температури внутрішніх органів у медичних процедурах або оперативній хірургії.
   - Харчова промисловість: Контроль температури під час процесу кулінарії, зберігання або транспортування харчових продуктів.

- Наукові дослідження: Вимірювання температури у лабораторних умовах або експериментах, де потрібна висока точність і стабільність. Це лише кілька прикладів застосування різних методів вимірювання температури, і реальні можливості залежатимуть від конкретних вимог вашого проекту та доступних технологій [30].

ВИСНОВКИ ДО РОЗДІЛУ 1

1. Розглянуто системи віддаленого контролю температури та проведено огляд різних методів вимірювання температури. Досліджено різні типи датчиків вимірювання температури та визначено їх особливості.
2. Визначено поняття "системи віддаленого контролю температури" та описано їх переваги та недоліки.
3. Проаналізовані різні типи датчиків вимірювання температури та їх особливості, обраний датчик для проекту - DS18B20.
4. Проведено огляд методів вимірювання температури, визначено їх переваги та недоліки.

РОЗДІЛ 2. КОНСТРУКЦІЯ СИСТЕМИ ВІДДАЛЕНОГО КОНТРОЛЮ ТЕМПЕРАТУРИ ТА ВИБІР СКЛАДОВИХ

2.1 Огляд основних компонентів системи:

Цей розділ присвячений детальному огляду основних компонентів системи віддаленого контролю температури. У цьому розділі розглядаються ключові компоненти, які використовуються в нашому проекті, включаючи Ардуіно Uno, компорт, датчик вимірювання температури DS18B20, дисплей LM016L та реле G5Q-1-DC5. Кожен компонент грає важливу роль в системі і вимагає окремого розгляду.

Також надається загальний огляд основних компонентів і пояснюється їх роль у системі віддаленого контролю температури. Розглядаються особливості кожного компонента та їх взаємодія з іншими елементами системи.

Цей розділ надасть повну інформацію про використані компоненти і допоможе краще зрозуміти їх функціонал та значення в контексті системи віддаленого контролю температури. Далі розглядається кожен компонент, його технічні характеристики, способи підключення та особливості використання.

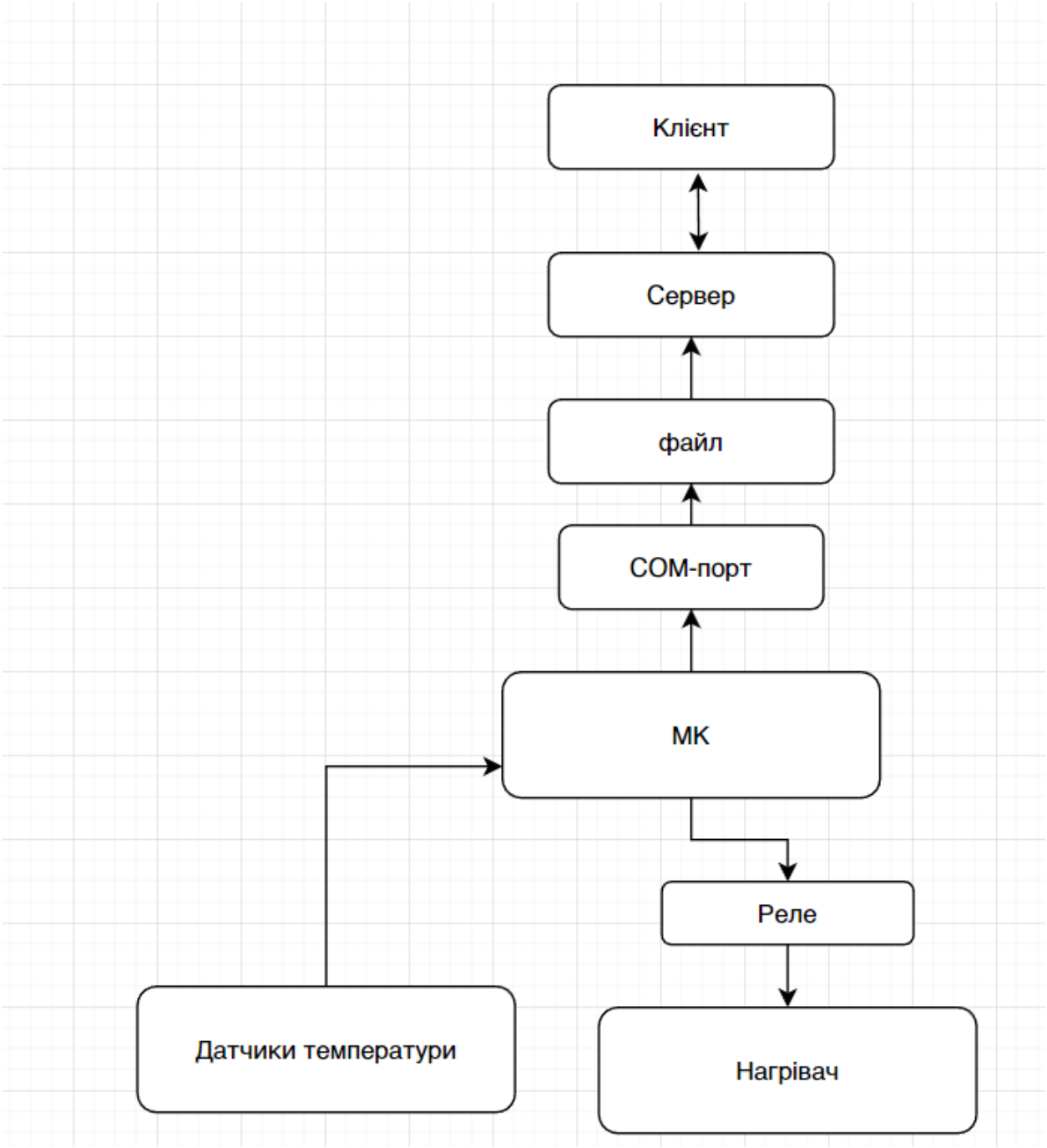

Рис. 2. Структурна схема об’єктів

На рис.2 зображена структурна схема об’єктів, де мікроконтролер отримує сигнали від датчиків температури, які підключені до його вхідних портів. Мікроконтролер перетворює ці сигнали у числовий формат та порівнює їх з заданим значенням температури. Після цього мікроконтролер записує дані про температуру в файл, а потім відправляє цей файл на сервер через COM-порт .

Сервер приймає ці дані. Користувач може отримати доступ до цих даних та керувати нагрівачами через завдяки реле.
Отже, зв'язок між датчиками, мікроконтролером та сервером є ключовим для правильної роботи системи контролю температури. Кожен з цих об'єктів виконує свою роль у цьому процесі та залежить від інших для забезпечення правильної та ефективної роботи системи.

2.1.1 Огляд Arduino Uno.

Ардуіно Uno є однією з найпопулярніших моделей платформ для створення прототипів в серії Ардуіно. Вона базується на мікросхемі ATmega328P і пропонує широкі можливості для розвитку інтерактивних проектів [10].

Ардуіно Uno має компактний розмір та простоту використання, що робить його ідеальним вибором для початківців і досвідчених розробників. Цей мікроконтролер має 14 цифрових входів/виходів, з яких 6 можуть використовуватися як ШІМ-виходи, 6 аналогових входів, кварцовий резонатор на частоті 16 МГц, вбудований USB-порт для зручного програмування та зв'язку з комп'ютером, а також ICSP-роз'єм для програмування мікроконтролера.

Ардуіно Uno працює з Arduino IDE - середовищем розробки, що надає зручність у програмуванні. Інтегрована розробка програмного забезпечення дозволяє розробникам створювати складні проекти швидко та ефективно. Багато розширень і модулів доступні для Ардуіно Uno, що дозволяє розширити його функціональність і використовувати його для різноманітних проектів.

Ардуіно Uno є потужним та гнучким мікроконтролером, який відкриває широкі можливості для реалізації вашої системи віддаленого контролю температури. Використовуючи Ардуіно Uno, ви можете програмувати різноманітні функції та забезпечити взаємодію з іншими компонентами

вашої системи, створюючи потужну та ефективну систему контролю температури.

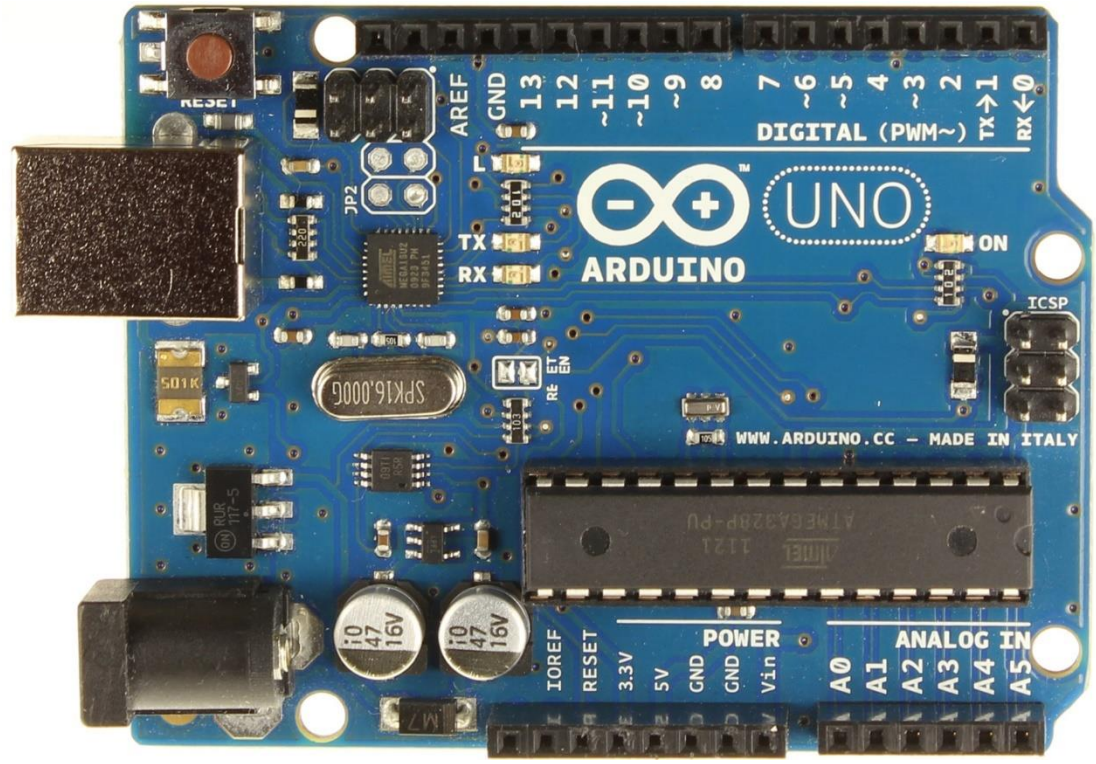

Рис. 2.1. Arduino Uno

Ардуіно Uno має кілька унікальних переваг, які роблять його популярним серед розробників та електронних ентузіастів:

Переваги:

1. Простота в використанні: Ардуіно Uno має простий інтерфейс та легкий у використанні синтаксис програмування, що дозволяє швидко розпочати роботу навіть початківцям.

2. Велика спільнота: Існує широка та активна спільнота розробників Ардуіно, яка надає підтримку, поради та ідеї через форуми, блоги та соціальні мережі.

3. Багатофункціональність: Ардуіно Uno має значну кількість вхідних та вихідних портів, що дозволяє підключати різні пристрої та модулі для розширення можливостей проекту.

4. Низька вартість: Ардуіно Uno є відносно доступним варіантом для електронних проектів, що дозволяє економити бюджет при розробці.

5. Кросплатформенність: Ардуіно Uno підтримується на різних операційних системах, таких як Windows, macOS та Linux, що забезпечує його універсальність та зручність в роботі.

Незважаючи на свої переваги, Ардуіно Uno також має кілька недоліків, які варто враховувати:

Недоліки:

1. Обмежені ресурси: У порівнянні з більш потужними мікроконтролерами, Ардуіно Uno має обмежену кількість пам'яті та обчислювальну потужність, що може обмежити складність проектів.

2. Обмеженість входів/виходів: Хоча Ардуіно Uno має декілька вхідних та вихідних портів

, їх кількість може бути недостатньою для деяких складних проектів, які вимагають більшого числа підключених пристроїв.

Завдяки своїй простоті в використанні, багатофункціональності та доступності, Ардуіно Uno став популярним вибором для розробників та ентузіастів, які бажають реалізувати свої ідеї у світі електроніки та автоматизації.

Arduino Uno має наступні характеристики:

1. Мікроконтролер: Arduino Uno використовує мікроконтролер ATmega328P, який працює на частоті 16 МГц і має 32 КБ флеш-пам'яті для зберігання програмного коду.

2. Вхідні/вихідні порти: Цей мікроконтролер має 14 цифрових вхідно-вихідних портів, з яких 6 можуть бути налаштовані як вихідні порти ШИМ (Pulse Width Modulation). Також є 6 аналогових вхідних портів.

3. Напруга живлення: Arduino Uno працює з напругою живлення 5 Вольт, яка може бути подана через USB-порт або зовнішнє джерело живлення.

4. Пам'ять: Мікроконтролер ATmega328P має 2 КБ оперативної пам'яті (SRAM) і 32 КБ флеш-пам'яті для зберігання програмного коду. Також є 1 КБ EEPROM для зберігання постійних даних.

5. Комунікаційні інтерфейси: Arduino Uno має вбудований USB-інтерфейс для зв'язку з комп'ютером або іншими пристроями. Також підтримуються інтерфейси UART, SPI (Serial Peripheral Interface) та I2C (Inter-Integrated Circuit) для зв'язку з іншими пристроями.

6. Роз'єми: Arduino Uno має роз'єми для підключення додаткових модулів та пристроїв, включаючи цифрові, аналогові та спеціальні роз'єми, такі як ICSP для програмування мікроконтролера.

2.1.2 Опис компорту.

Компорт (Serial Port) є важливим елементом зв'язку для Arduino Uno в системі вимірювання температури в приміщенні. Він забезпечує можливість обміну даними між Arduino Uno та іншими пристроями.

Arduino Uno має вбудований апаратний Serial Port, який підтримує TTL-рівні напруги (Transistor-Transistor Logic). Це означає, що для зв'язку з комп'ютером або іншими пристроями потрібно використовувати перехідник рівнів напруги для перетворення TTL-сигналу на відповідний рівень напруги, який підтримується пристроєм [11].

Компорт Arduino Uno підтримує послідовний (серійний) зв'язок за допомогою UART (Universal Asynchronous Receiver-Transmitter) протоколу. Це дозволяє передавати та отримувати дані в режимі реального часу зі швидкістю до 115200 біт на секунду. Компорт може бути використаний для зв'язку з комп'ютером для завантаження програмного коду на Arduino Uno або для обміну даними з іншими пристроями через послідовний зв'язок.

Для взаємодії з компортом Arduino Uno використовується бібліотека SoftwareSerial або вбудований апаратний Serial Port, залежно від потреб проекту. Бібліотека SoftwareSerial дозволяє створювати додаткові віртуальні компорти, що розширює можливості зв'язку з іншими пристроями, коли всі апаратні Serial Ports використовуються.

Компорт є важливою складовою Arduino Uno, яка дозволяє забезпечити зв'язок з іншими пристроями та обмін даними, що розширює можливості платформи Arduino Uno для створення різноманітних проектів та забезпечення в заємодії з зовнішнім світом.

Рис. 2.2. Serial Ports

Характеристики компорту Arduino Uno:

1. Інтерфейс: Компорт Arduino Uno підтримує послідовний (серійний) зв'язок через UART протокол.

2. Швидкість передачі даних: Швидкість передачі даних може досягати значення до 115200 біт на секунду.

3. Рівень напруги: Компорт працює з TTL-рівнями напруги (Transistor-Transistor Logic).

4. Підтримка вбудованого Serial Port: Arduino Uno має вбудований апаратний Serial Port для зручного з'єднання з комп'ютером або іншими пристроями.

5. Підтримка SoftwareSerial: Є можливість використовувати бібліотеку SoftwareSerial для створення додаткових віртуальних компортів.

Переваги компорту Arduino Uno:

1. Універсальність: Компорт Arduino Uno забезпечує зв'язок з різноманітними пристроями, включаючи комп'ютери, мікроконтролери, модулі та інші зовнішні пристрої.

2. Простота використання: Взаємодія з компортом Arduino Uno відбувається за допомогою послідовного зв'язку, що дозволяє легко передавати та отримувати дані з інших пристроїв.

3. Широкі можливості: Компорт підтримує швидкість передачі даних до 115200 біт на секунду, що забезпечує ефективну комунікацію з високою швидкістю.

4. Підтримка бібліотек: Компорт Arduino Uno підтримує різні бібліотеки, такі як SoftwareSerial, що розширюють його можливості зі зв'язку та взаємодії з іншими пристроями.

Недоліки компорту Arduino Uno:

1. Обмежена кількість компортів: Arduino Uno має обмежену кількість фізичних компортів, що може обмежити одночасний зв'язок з багатьма пристроями.

2. Вимога до перехідника рівнів напруги: Для зв'язку з комп'ютером або пристроями, які використовують відмінні від TTL-рівні напруги, може знадобитися використання перехідника рівнів напруги.

3. Обмеження швидкості передачі даних: Максимальна швидкість передачі даних 115200 біт на секунду може бути недостатньою для деяких вимог, особливо у великих мережах або проектах з високою швидкістю обміну даними.

Незважаючи на обмеження, компорт Arduino Uno все ще є потужним та зручним інтерфейсом для зв'язку та взаємодії з різними пристроями, забезпечуючи надійну комунікацію та можливість реалізації різноманітних проектів.

Компорт Arduino Uno знаходить широке застосування в різних сферах життя і проектах. Ось деякі приклади, де компорт може бути використаний:

1. Розробка електронних пристроїв: Компорт Arduino Uno є важливим елементом при розробці різноманітних електронних пристроїв, включаючи робототехніку, сенсорні системи, автоматизацію домашнього розуму, контроль та моніторинг систем, тощо.

2. Інтерактивні проекти: Arduino Uno дозволяє створювати інтерактивні проекти, такі як ігри, музичні інструменти, арт-інсталяції, світлові шоу та багато іншого. Завдяки можливості зв'язку з комп'ютером або іншими пристроями, Arduino Uno може служити як основа для взаємодії з різними сенсорами та виконання визначених дій.

3. Навчання електроніки та програмування: Компорт Arduino Uno використовується в навчальних закладах та навчальних курсах для введення студентів у світ електроніки та програмування. Його легкість використання та широкі можливості дозволяють студентам зробити практичні проекти та експерименти з реальними пристроями.

4. DIY-проекти та хобі: Arduino Uno дуже популярний серед ентузіастів DIY (Do-It-Yourself) та хобістів. Він дозволяє створювати власні проекти, реалізовувати ідеї та вдосконалювати різноманітні пристрої, починаючи від домашньої автоматизації та контролю побутових пристроїв до побудови роботів та інтерактивних іграшок.

5. Промислові застосування: Компорт Arduino Uno також може бути використаний у промислових проектах, де потрібна невелика автоматизація, моніторинг чи керування. Він може бути частиною систем контролю та моніторингу, включаючи системи управління температурою, освітленням, системи безпеки та багато інших.

2.1.3 Опис датчика вимірювання температури ds18b2.

Датчик DS18B20 є одним з популярних цифрових датчиків температури, [12] який використовує шину однозв'язкового інтерфейсу (OneWire) для передачі даних.

Рис. 2.3. Датчик температури DS18B20 для Arduino

Ось деякі характеристики цього датчика:

- Діапазон вимірювання температури: DS18B20 може вимірювати температуру в діапазоні від -55°C до +125°C. Цей широкий діапазон робить його придатним для багатьох застосувань, як в холодних, так і високотемпературних середовищах.

- Висока точність: DS18B20 має високу точність вимірювання температури з роздільною здатністю до 0,0625°C. Це дозволяє отримувати досить точні вимірювання температури для багатьох додатків.

- Цифровий інтерфейс: Датчик DS18B20 використовує цифровий інтерфейс, що дозволяє підключати його до мікроконтролерів та інших пристроїв з цифровим вхідно-вихідним портом.

- Унікальна адреса: Кожен датчик DS18B20 має унікальну 64-бітну адресу, що дозволяє підключати багато датчиків до однієї шини, забезпечуючи можливість мультиточкового вимірювання температури.

- Легке програмування: DS18B20 має широку підтримку в програмному забезпеченні, що дозволяє легко програмувати зчитування температури та використовувати його дані для подальшої обробки.

Переваги датчика DS18B20:

1. Цифровий інтерфейс: Його цифровий інтерфейс забезпечує легку інтеграцію з різними мікроконтролерами та пристроями.

2. Висока точність: Висока точність вимірювання температури дозволяє отримувати надійні та точні результати.

3. Гнучкість у підключенні: Можливість підключати багато датчиків до однієї шини дозволяє вимірювати температуру в різних точках або приміщеннях.

4. Широкий діапазон вимірювання: Великий діапазон вимірювання температури дозволяє використовувати датчик у різних середовищах та додатках.

Недоліки датчика DS18B20:

1. Потребує програмування: Для використання датчика DS18B20 потрібно програмувати мікроконтролер або інший пристрій для зчитування даних з датчика.

2. Обмежена швидкість оновлення: Датчик DS18B20 має обмежену швидкість оновлення, тому не підходить для додатків, які вимагають високої частоти зчитування температури.

Використання датчика DS18B20:

Датчик DS18B20 знайшов широке застосування в різних галузях, включаючи:

- Системи контролю температури: Датчик DS18B20 може використовуватись у системах контролю температури, які забезпечують стабільну температуру в приміщеннях, холодильних камерах, акваріумах тощо.

- Автоматизовані системи: DS18B20 може використовуватися у автоматизованих системах, таких як системи контролю освітлення, системи управління HVAC (опалення, вентиляція, кондиціювання повітря) та системи безпеки.

- Моніторинг температури: Датчик DS18B20 може бути використаний для моніторингу температури в різних середовищах, включаючи промислові об'єкти, складські приміщення, тепличні установки тощо.

- Проектування пристроїв: DS18B20 може бути використаний у проектуванні різноманітних пристроїв, таких як погодні станції, системи моніторингу, пристрої автоматизації та багато інших.

Таким чином, датчик вимірювання температури DS18B20 є надійним і популярним рішенням для вимірювання температури у різних додатках і галузях. Він має високу точність, гнучкість у підключенні та широкі можливості застосування.

2.1.4 Опис дисплею LM016l.

Дисплей LM016L є текстовим LCD-дисплеєм, який використовується для відображення інформації у системі віддаленого контролю температури. Завдяки своїй простоті використання та низькій вартості, він є популярним вибором для відображення різних значень та повідомлень [13].

Рис. 2.4. Дисплей LM016L

Дисплей LM016L, який є одним з популярних символьних LCD дисплеїв. Ось деякі характеристики цього дисплея [27]:

- Розмір: Дисплей LM016L має 2 рядки по 16 символів, що дозволяє відображати текст та символи у зручний спосіб.

- Тип виводів: Дисплей має паралельний інтерфейс, який забезпечує просте підключення до мікроконтролера або іншого пристрою.

- Контрастність: Дисплей LM016L має можливість налаштування контрастності, що дозволяє підлаштовувати його під потреби та умови освітлення.

- Джерело живлення: Дисплей живиться від джерела живлення 5 Вольт, що робить його сумісним з багатьма мікроконтролерами та пристроями.

- Простий у використанні: Дисплей має простий інтерфейс команд, які дозволяють виводити текст, символи та здійснювати інші дії з відображенням.

Переваги дисплея LM016L:

1. Чітке відображення: Дисплей LM016L забезпечує чітке та яскраве відображення тексту та символів, що полегшує читання та сприяє зручності використання.

2. Просте підключення: Завдяки паралельному інтерфейсу, підключення дисплея LM016L до мікроконтролера або іншого пристрою є простим та зручним.

3. Гнучкість у використанні: Дисплей може відображати різні символи, текст та графічні елементи, що дозволяє створювати різноманітні інтерфейси та повідомлення.

Недоліки дисплея LM016L:

1. Обмежений розмір: Дисплей має обмежений розмір 2 рядків по 16 символів, що може бути недостатньо для деяких додатків, які вимагають більшого простору для відображення інформації.

2. Відсутність підсвітки: Деякі версії дисплея LM016L можуть не мати вбудованої підсвітки, що може ускладнити читання в умовах обмеженого освітлення.

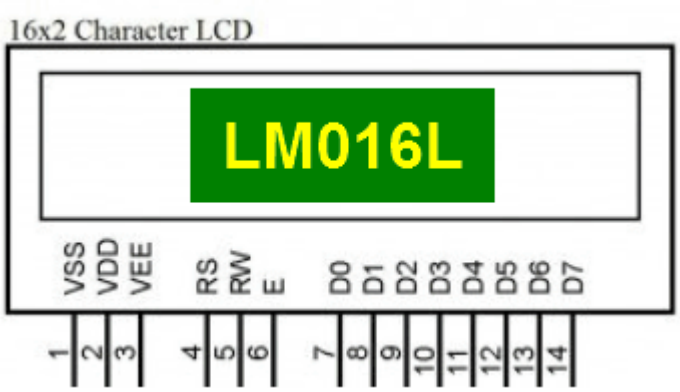

Рис. 2.5. Піни дисплею LM016L

Застосування дисплея LM016L:

Дисплей LM016L знаходить своє застосування в різних галузях, включаючи:

- Електроніка та робототехніка: Дисплей може бути використаний для відображення інформації та стану пристроїв, таких як роботи, контролери, модулі вимірювання та багато інших.

- Автоматизація: Дисплей LM016L може бути використаний у системах автоматизації для відображення стану процесів, налаштування параметрів та інформування операторів.

- Контроль та моніторинг: Дисплей може використовуватись для відображення даних з датчиків, стану систем контролю температури, вологості, тиску тощо.

Таким чином, дисплей LM016L є зручним та популярним рішенням для відображення інформації в різних пристроях та додатках. Він має чітке відображення, просте підключення та гнучкість у використанні, що дозволяє його використовувати у багатьох сферах та проектах..

2.1.5 Опис реле G5Q-1-DC5.

Реле G5Q-1-DC5 є електромеханічним реле з одним контактом і сприятливою для середовища котушкою постійного струму 5 Вольт. [14] Воно використовується для керування електричними навантаженнями на основі сигналів, отриманих від мікроконтролера або іншого електронного пристрою.

Рис. 2.5. Реле G5Q-1-DC5

Характеристики реле G5Q-1-DC5 включають:

- Напруга котушки: 5 Вольт DC

- Максимальний струм котушки: 26.7 мА

- Контактна конфігурація: 1 перехідний контакт (SPDT)

- Максимальний комутаційний струм: 10 А (при 250 Вольт AC)

- Максимальна комутаційна напруга: 250 Вольт AC

- Електричне життя: 100 000 циклів комутації

Переваги реле G5Q-1-DC5 [28]:

1. Надійність: Реле G5Q-1-DC5 має високу надійність і стабільну роботу, що дозволяє йому успішно керувати електричними навантаженнями протягом тривалого часу.

2. Компактний розмір: Реле має компактну конструкцію, що дозволяє легко розмістити його в пристроях з обмеженим простором.

3. Широкий діапазон комутації: Реле може комутувати струми до 10 А при 250 Вольт AC, що робить його відповідним для використання з різними електричними навантаженнями.

Недоліки реле G5Q-1-DC5:

1. Обмежена напруга котушки: Реле працює з напругою котушки 5 Вольт DC, що може бути несумісним з деякими джерелами живлення.

2. Обмежений контактний струм: Максимальний комутаційний струм реле становить 10 А, що може бути недостатнім для великих електричних навантажень.

Застосування реле G5Q-1-DC5:

- Автоматизація та контроль: Реле може бути використане для керування різними електричними пристроями, системами контролю, автоматизованими процесами і системами.

- Електроніка та робототехніка: Реле G5Q-1-DC5 може бути використане в різних електронних проектах, роботах та системах, де потрібно керувати електричними навантаженнями.

- Електромонтаж та електрика: Реле може бути використане в електромонтажних роботах, розетках, електричних панелях та інших системах електропостачання.

Реле G5Q-1-DC5 є потужним і надійним елементом для керування електричними навантаженнями в різних пристроях і системах. Його компактні розміри, висока надійність та широкий діапазон комутації роблять його відмінним вибором для багатьох застосувань.

2.2 ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ. СЕРЕДОВИЩЕ РОЗРОБКИ ARDUINO IDE

2.2.1 Arduino IDE

ARDUINO IDE (Integrated Development Environment) є офіційним середовищем розробки для програмування платформи Arduino. Воно надає зручний інтерфейс та інструменти для створення, перевірки та завантаження програмного коду на плату Arduino.[15]

Рис. 2.6. Інтерфейс Arduino IDE

Основні характеристики ARDUINO IDE:

1. Синтаксичний підсвітка та автодоповнення: ARDUINO IDE надає можливість виділяти синтаксичні елементи мови програмування, що полегшує розуміння коду. Також воно автоматично доповнює код на основі наявних функцій та змінних.
2. Вбудована бібліотека: ARDUINO IDE має велику кількість вбудованих бібліотек, які містять готові функції та процедури для різних операцій, включаючи роботу з платами розширення, зчитування даних з датчиків, управління виводами та багато іншого.

3. Зручний монітор послідовного порту: ARDUINO IDE має вбудований монітор послідовного порту, який дозволяє переглядати та відлагоджувати дані, що передаються між Arduino та комп'ютером через послідовний порт. Це дозволяє контролювати вхідні та вихідні дані та виявляти помилки.
4. Простота використання: ARDUINO IDE має простий та інтуїтивно зрозумілий інтерфейс, що дозволяє швидко почати розробку проектів з використанням Arduino. Воно підтримує як початківців, так і досвідчених розробників.
5. Платформонезалежність: ARDUINO IDE підтримує роботу на різних операційних системах, включаючи Windows, macOS та Linux. Це дозволяє розробляти проекти на Arduino незалежно від використовувано ї платформи.
ARDUINO IDE є популярним інструментом серед спільноти розробників Arduino. Його простота використання, вбудовані можливості та багатий функціонал роблять його ідеальним вибором для створення та програмування проектів з використанням Arduino.

2.2.2 Огляд SPLAN

Splan є потужним середовищем розробки, спеціально створеним для проектування електричних схем та печатних плат. Воно надає розширені можливості для створення високоякісних схем та печатних плат з точністю до деталі. [16]

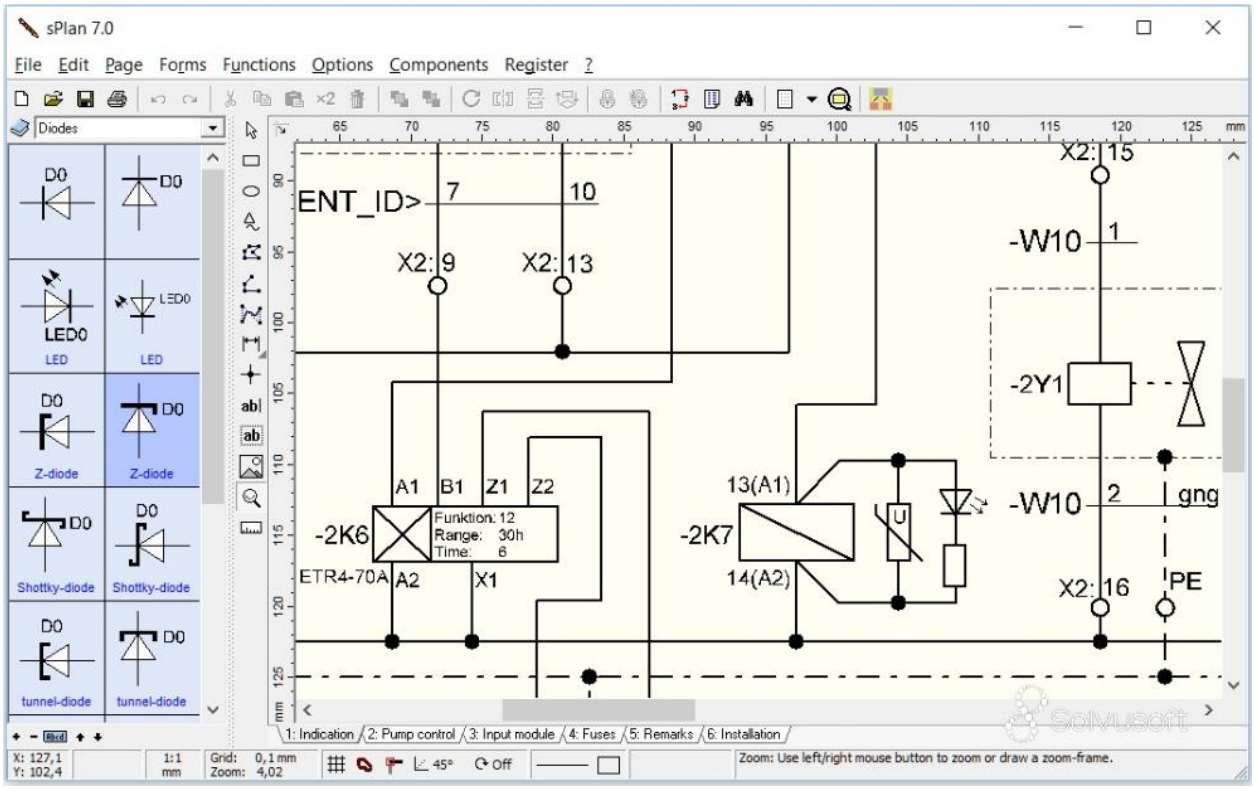

Рис. 2.6. Інтерфейс sPlan 7.0

Основні характеристики Splan:

1. Велика бібліотека компонентів: Splan має широкий вибір вбудованих компонентів, які можна використовувати для побудови електричних схем. Вона включає різноманітні електронні компоненти, з'єднувачі, джерела живлення та інші важливі елементи.

2. Простота використання: Splan має зручний інтерфейс, що дозволяє легко створювати, редагувати та маніпулювати елементами схеми. Він пропонує інтуїтивно зрозумілі інструменти для додавання компонентів, з'єднання виводів та налаштування параметрів.

3. Можливість симуляції: Splan дозволяє симулювати роботу електричних схем перед їх фізичною реалізацією. Це дозволяє виявляти помилки та перевіряти працездатність схеми перед початком виробництва.

4. Гнучкість в розміщенні компонентів: Splan дозволяє гнучко розміщувати компоненти на схемі та вільно маніпулювати їх положенням. Це дозволяє оптимізувати розташування компонентів, забезпечуючи компактність та зручність монтажу.

5. Експорт та друк схем: Splan надає можливість експортувати схеми у різних форматах, таких як PDF, зображення (JPG, PNG) або документи формату SVG. Ви також можете без проблем друкувати схеми без втрати якості. Splan є незамінним інструментом для розробки електричних схем та печатних плат. Використовуючи його можливості, ви можете ефективно проектувати та виробляти якісні електронні пристрої.

2.2.3 Огляд PROTEUS 8 PROFESSIONAL

Proteus 8 Professional є потужним інструментом для симуляції, проектування та віртуального тестування електронних пристроїв. Він забезпечує розширені можливості для моделювання роботи електронних схем та перевірки їх працездатності. [17]

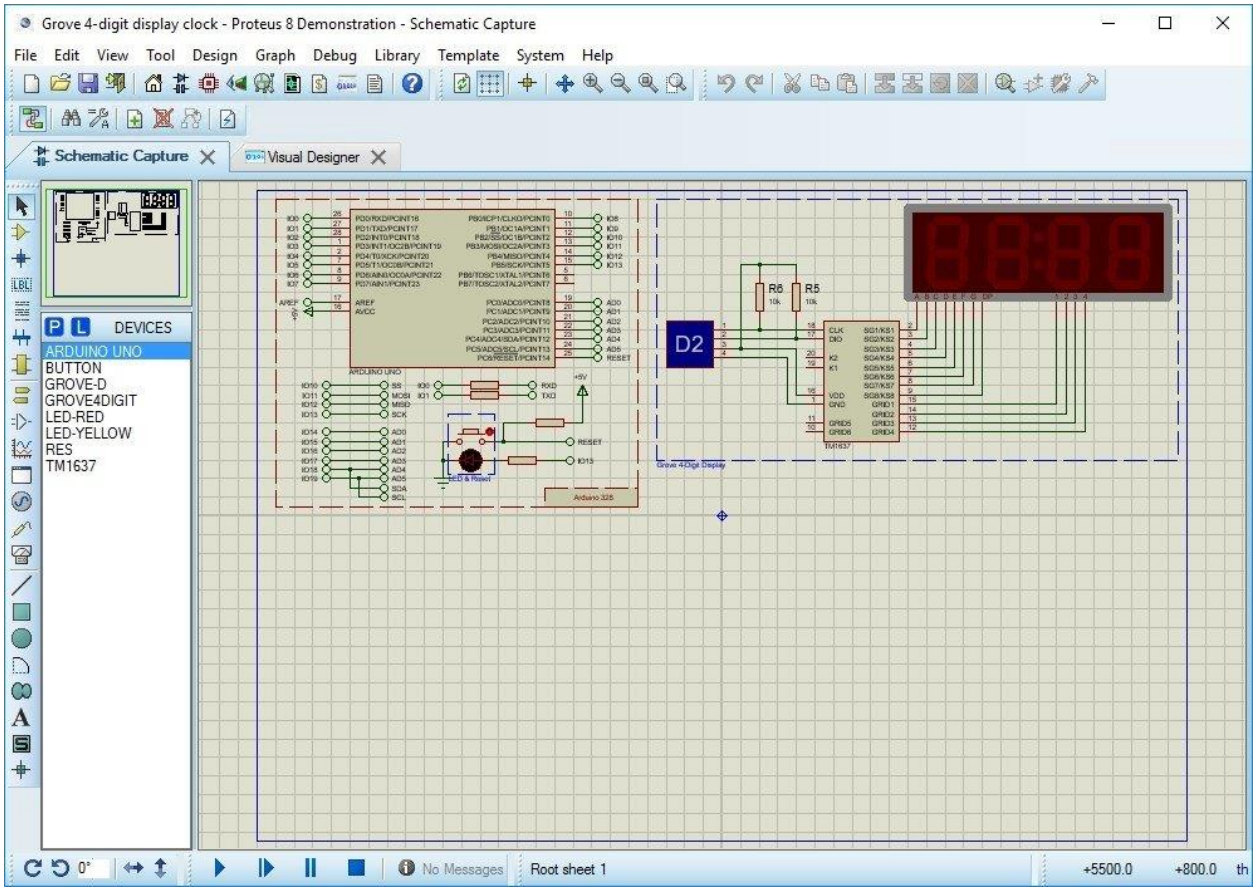

Рис. 2.7. Інтерфейс Proteus 8 Professional

Основні характеристики Proteus 8 Professional:

1. Симуляція електронних схем: Proteus дозволяє симулювати роботу електронних схем перед їх фізичною реалізацією. Ви можете перевірити

працездатність схеми, провести тестування та оптимізувати її перед виробництвом.

2. Велика бібліотека компонентів: Proteus має широкий вибір вбудованих компонентів, що дозволяє легко створювати складні електронні схеми. Бібліотека включає різноманітні електронні компоненти, мікроконтролери, сенсори, актуатори та багато інших.

3. Зручне редагування схем: Proteus пропонує зручний інтерфейс для редагування електронних схем. Ви можете легко додавати та змінювати компоненти, з'єднувати їх виводи, налаштовувати параметри та встановлювати залежності між ними.

4. Моделювання роботи мікроконтролерів: Proteus підтримує моделювання роботи мікроконтролерів, що дозволяє вам перевірити програмне забезпечення до його фізичної реалізації. Ви можете встановлювати брейкпоінти, відлагоджувати код та тестувати його працездатність.

5. Віртуальна плата для тестування: Proteus має можливість створювати віртуальну плату для тестування інтерфейсів та зв'язку між компонентами. Ви можете проводити віртуальні тести на реальному обладнанні та перевіряти його працездатність.

Переваги Proteus 8 Professional:

1. Потужність симуляції: Proteus надає високу точність та реалістичність симуляції електронних схем, що дозволяє вам віртуально перевірити працездатність пристрою перед фізичною реалізацією.

2. Широка бібліотека компонентів: Велика кількість вбудованих компонентів у бібліотеці Proteus дозволяє швидко та зручно проектувати складні електронні схеми, що економить час та зусилля при розробці проекту.

3. Зручне редагування схем: Інтерфейс Proteus забезпечує зручне редагування електронних схем, що спрощує процес розробки та модифікації схеми. Ви можете легко додавати, змінювати та з'єднувати компоненти, налаштовувати їх параметри та залежності.

4. Моделювання мікроконтролерів: Proteus підтримує моделювання роботи мікроконтролерів, що дозволяє вам перевірити програмне забезпечення до його фізичної реалізації. Ви можете встановлювати брейкпоінти, відлагоджувати код та тестувати його працездатність.
5. Віртуальна плата для тестування: Proteus дозволяє створювати віртуальну плату для тестування функціональності інтерфейсів та з'єднань між компонентами. Це дозволяє проводити віртуальні тести та відлагоджувати пристрій без необхідності фізичної плати.

Недоліки Proteus 8 Professional:

1. Вартість ліцензії: Proteus 8 Professional є комерційним програмним забезпеченням, тому його ліцензія може бути високою для окремих користувачів або невеликих команд розробників.
2. Вимоги до апаратного забезпечення: Для оптимальної роботи Proteus 8 Professional потрібний потужний комп'ютер з достатньою кількістю оперативної пам'яті та швидким процесором. Недостатні ресурси комп'ютера можуть призвести до повільної роботи програми або неправильної симуляції.
3. Відсутність деяких спеціалізованих компонентів: Незважаючи на широку бібліотеку компонентів, деякі спеціалізовані компоненти можуть бути відсутніми. У такому випадку може знадобитися створення власного компонента або пошук альтернативних рішень.
4. Навчання використанню програми: Proteus 8 Professional має досить складний інтерфейс, що може вимагати часу та зусиль для оволодіння всіма його функціями та можливостями. Необхідно провести достатньо часу на навчання та практику для ефективного використання програми.

Proteus 8 Professional - потужний інструмент для симуляції електронних схем, що надає високу точність та реалістичність симуляції, широку бібліотеку компонентів, зручне редагування схем та можливість моделювання мікроконтролерів. Він також дозволяє створювати віртуальну плату для тестування функціональності та з'єднань. Проте, варто враховувати вартість

ліцензії, вимоги до апаратного забезпечення, відсутність деяких спеціалізованих компонентів та необхідність вивчення програми для ефективного використання.

2.2.4 Огляд Visual Studio

Visual Studio є однією з найпопулярніших інтегрованих середовищ розробки (IDE) для програмування різних мов, включаючи мови програмування, які використовуються для розробки систем віддаленого контролю температури [18].

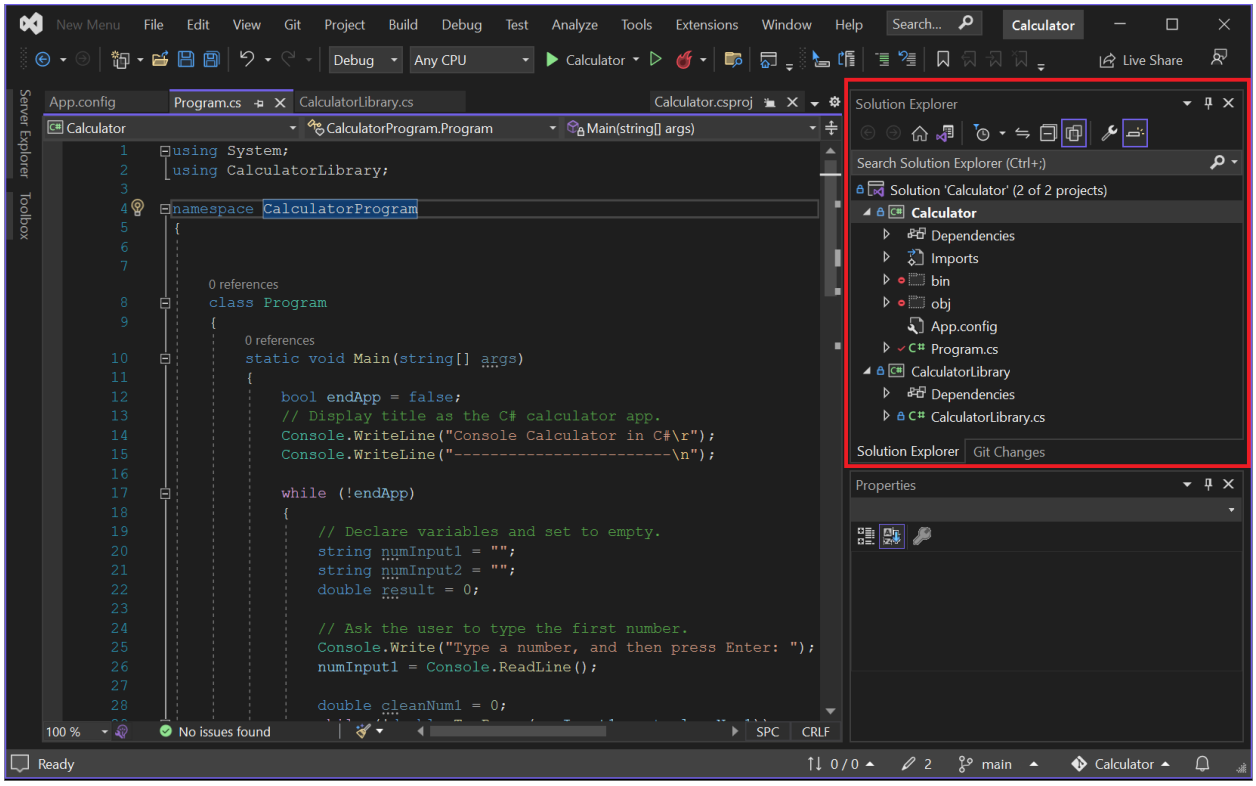

Рис. 2.8. Visual Studio

Деякі характеристики Visual Studio включають:

1. Розширена підтримка мов програмування: Visual Studio надає розширену підтримку для різних мов програмування, таких як C++, C#, Visual Basic та інші. Це дозволяє розробникам використовувати їх у своїх проектах з високим рівнем зручності та продуктивності.
2. Інтегровані інструменти розробки: Visual Studio має широкий набір інтегрованих інструментів розробки, таких як редактор коду з підсвіткою

синтаксису, автодоповнення, налагоджувач, система керування версіями та багато інших. Ці інструменти допомагають розробникам ефективно працювати над проектами та підвищують продуктивність розробки.

3. Широка спільнота та підтримка: Visual Studio має велику спільноту розробників, яка надає допомогу, поради та рішення проблем у випадку потреби. Крім того, Microsoft, розробник Visual Studio, надає активну підтримку та оновлення продукту, забезпечуючи стабільну роботу та нові функціональні можливості.

4. Розширюваність: Visual Studio є дуже розширюваним, що дозволяє використовувати сторонні розширення та плагіни для розширення функціональності IDE. Це дозволяє розробникам налаштовувати робоче середовище під свої потреби та використовувати додаткові інструменти та розширення.

Переваги Visual Studio включають багатомовну підтримку, потужність інтегрованих інструментів, широку спільноту та підтримку, а також можливість розширення. Проте, деякі недоліки включають значні вимоги до ресурсів комп'ютера та необхідність навчання та оволодіння середовищем розробки для ефективного використання всіх його можливостей.

Visual Studio використовується в різних сферах розробки програмного забезпечення і додатків.

Ось декілька прикладів його застосування:

1. Розробка десктопних додатків: Visual Studio дозволяє розробляти десктопні додатки для операційних систем Windows, включаючи програми, інструменти, утиліти та інші програмні продукти.

2. Веб-розробка: Visual Studio надає інструменти для розробки веб-додатків та сайтів, включаючи можливості роботи з HTML, CSS, JavaScript, ASP.NET та іншими технологіями.

3. Розробка мобільних додатків: Visual Studio підтримує розробку мобільних додатків для платформ Android та iOS, включаючи інтеграцію з популярними фреймворками, такими як Xamarin.
4. Ігрова розробка: Visual Studio має функціональність для розробки ігор, включаючи підтримку популярних гейм-движків, таких як Unity та Unreal Engine.
5. Розробка хмарних додатків: Visual Studio надає інструменти для розробки хмарних додатків, включаючи роботу з платформами Azure та AWS.
6. Розробка інтерфейсів користувача: Visual Studio має засоби для розробки графічних інтерфейсів користувача, таких як Windows Forms та WPF, що дозволяють створювати різноманітні програми зі зручним інтерфейсом.
Це лише кілька прикладів сфер, де можна використовувати Visual Studio. Його потужність та гнучкість дозволяють розробляти програмне забезпечення для різних потреб і галузей.
Visual Studio - потужне інтегроване середовище розробки (IDE), яке надає розширену підтримку різних мов програмування, розширені інструменти розробки, широку спільноту та підтримку, а також можливість розширення. Воно є популярним вибором для розробки систем віддаленого контролю температури, забезпечуючи зручне та продуктивне середовище розробки. Проте, варто враховувати вимоги до ресурсів комп'ютера та необхідність навчання для ефективного використання програми.

2.2.5 Огляд Virtual Null Modem

Віртуальний нуль-модем, також відомий як Virtual Null Modem, є програмним рішенням, що дозволяє емулювати з'єднання між віртуальними портами COM на комп'ютері. Його основна функція полягає в передачі даних між програмами, які використовують ці віртуальні порти. Завдяки Virtual Null Modem розробники можуть тестувати та відлагоджувати програми, які взаємодіють з реальними пристроями через порти COM, без необхідності фізичного з'єднання. [19]

Це програмне рішення має кілька переваг. Воно надає віртуальне середовище для розробки, що дозволяє ефективно тестувати та відлагоджувати програмне забезпечення. Емуляція різних типів з'єднання, таких як прямий зв'язок, зв'язок через модем або зв'язок через мережу, розширює можливості тестування та дослідження. Крім того, Virtual Null Modem має зручний інтерфейс, що спрощує налаштування віртуальних портів COM та встановлення з'єднання між ними.

Проте, варто враховувати його обмежену функціональність. Віртуальний нуль-модем зосереджений на емуляції портів COM та передачі даних, і не надає розширених функцій для обробки сигналів та управління потоками даних. Також, він може бути менш ефективним у випадках, коли потрібно з'єднати багато пристроїв або використовувати його у великих мережах.

Virtual Null Modem знайшов своє застосування в різних сферах. Він використовується в процесі розробки та відлагодження програмного забезпечення, тестуванні пристроїв, моделюванні взаємодії між пристроями та у дослідницьких проектах. Також, його можна використовувати в навчальних цілях для демонстрації роботи портів COM та з'єднання між пристроями.

Основні переваги Virtual Null Modem включають:

1. Віртуальне середовище: Воно дозволяє розробникам тестувати та відлагоджувати програмне забезпечення, яке взаємодіє з реальними пристроями через порти COM, без фізичного підключення.
2. Емуляція зв'язку: Віртуальний нуль-модем дозволяє емулювати різні види з'єднання, такі як прямий зв'язок, зв'язок через модем або зв'язок через мережу.
3. Простота використання: Він має зручний інтерфейс та просту налаштування, що дозволяє швидко налаштувати віртуальні порти COM та встановити з'єднання між ними.

Незважаючи на переваги, Virtual Null Modem також має деякі недоліки:

1. Обмежена функціональність: Це програмне забезпечення спрямоване виключно на емуляцію портів COM і передачу даних, тому воно не надає розширених функцій для обробки сигналів, управління потоками даних та інших функцій, які можуть бути доступні на реальних пристроях.
2. Обмежена масштабованість: Віртуальний нуль-модем придатний для використання на одному комп'ютері, але може бути обмеженим для застосування у великих мережах або у випадках, коли потрібно з'єднувати велику кількість пристроїв.

Застосування Virtual Null Modem включає:

1. Розробка та відлагодження програмного забезпечення: Він дозволяє розробникам перевірити взаємодію програм з реальними пристроями, емулюючи з'єднання через віртуальні порти COM.
2. Тестування та діагностика обладнання: Він може бути використаний для тестування пристроїв, що взаємодіють через порти COM, дозволяючи віртуально підключити їх до тестових систем.
3. Симуляція з'єднання: Він може бути використаний для симуляції з'єднання через порти COM, дозволяючи емулювати певні сценарії взаємодії між пристроями для тестування та дослідження.

ВИСНОВКИ ДО 2 РОЗДІЛУ

1. Проведено детальний огляд основних компонентів системи та середовища розробки. В процесі огляду було здійснено аналіз і вибір компонентів, які є необхідними для реалізації системи віддаленого контролю температури.
2. Для реалізації системи обрано наступні компоненти: Arduino Uno як основну мікроконтролерну платформу, компорт для з'єднання з комп'ютером, датчик вимірювання температури DS18B20, дисплей LM016L для відображення температурних даних та реле G5Q-1-DC5 для керування зовнішніми пристроями на основі вимірювання температури.

3. Також проведено огляд середовища розробки, включаючи Arduino IDE, яке використовується для програмування мікроконтролерів Arduino, SPLAN - програмного забезпечення для проектування схем електричних мереж, Proteus 8 Professional - середовища для симуляції та верифікації електронних схем, а також Virtual Null Modem - інструменту для емуляції віртуальних послідовних портів.

РОЗДІЛ 3. ОПИС РОБОТИ ПРИСТРОЮ.

3.1 Моделювання системи віддаленого контролю температури в приміщенні в Proteus

Модель системи віддаленого контролю температури в приміщенні (Рис 3.1.1) складається з таких основних елементів [20]:

- Мікроконтролер Arduino Uno.
- Датчик температури DS18B20.
- Дисплей LM016L
- Реле G5Q-1-DC5
- Компорт
- Нагрівач який підключений до резистора R2 (Рис 3.1.2)

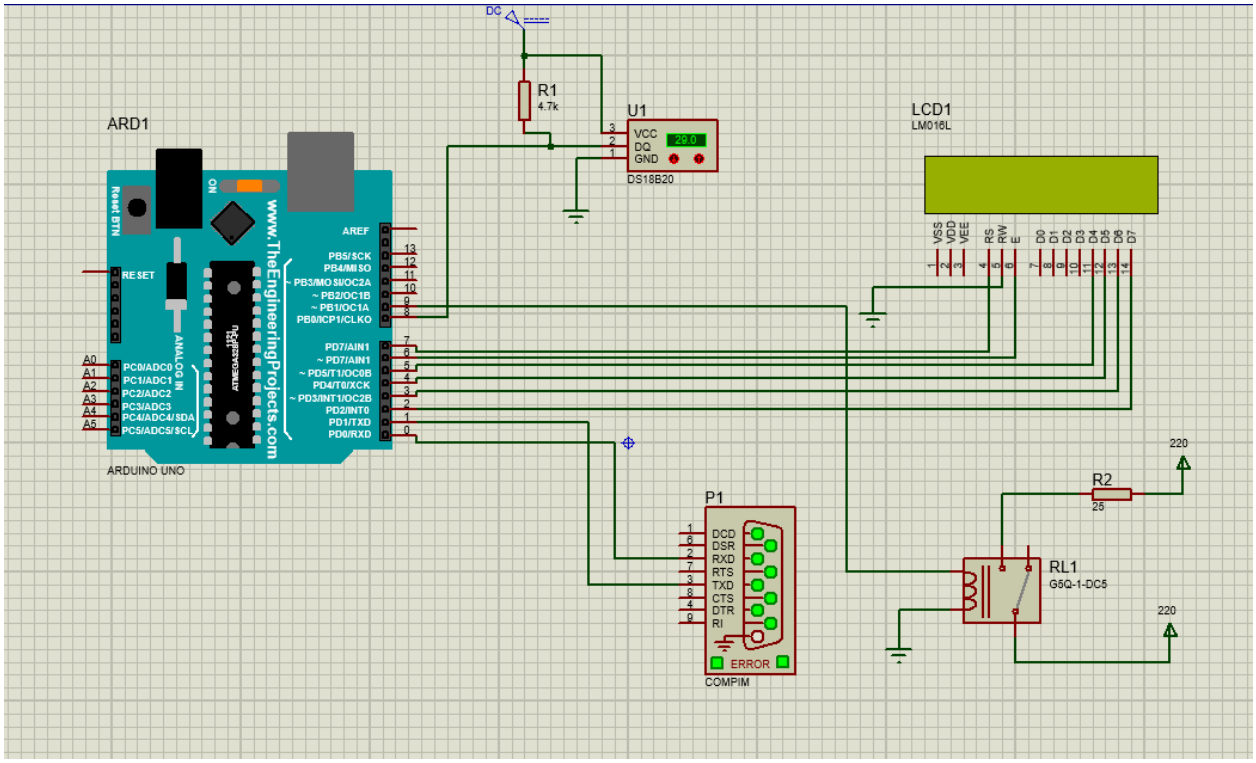

Рис 3.1.1

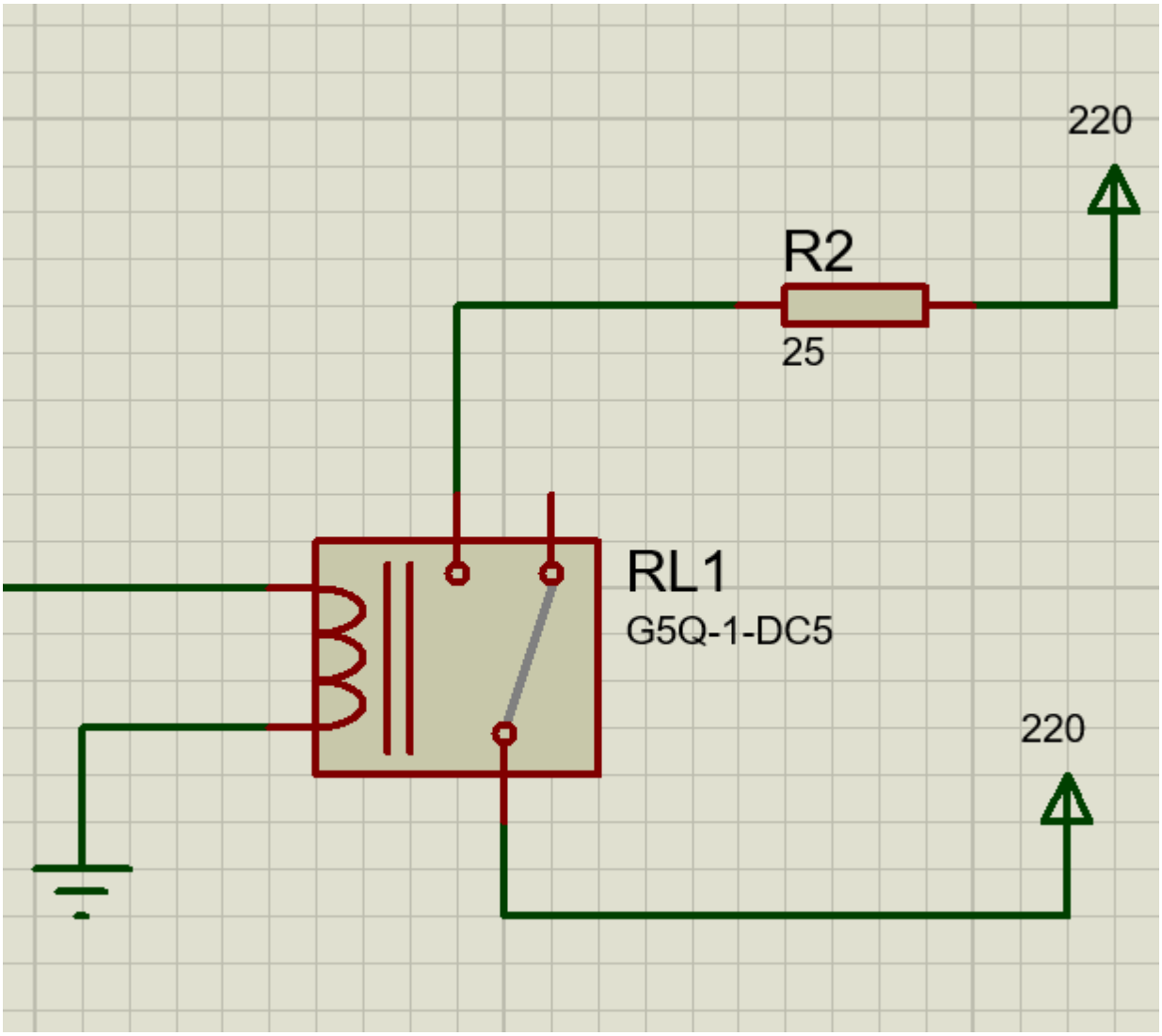

Рис. 3.1.2

На Рис 3.1.2 зображений резистор R2. Замість цього резистору можна під’єднати будь-який нагрівач потужністю не більше 2Квт та холодним опором не менше 25 Ом.

Живлення реле та нагрівача – 220В.

Схема системи віддаленого контролю температури живиться від 5В.

3.2 Розробка програмного забезпечення для Arduino IDE.

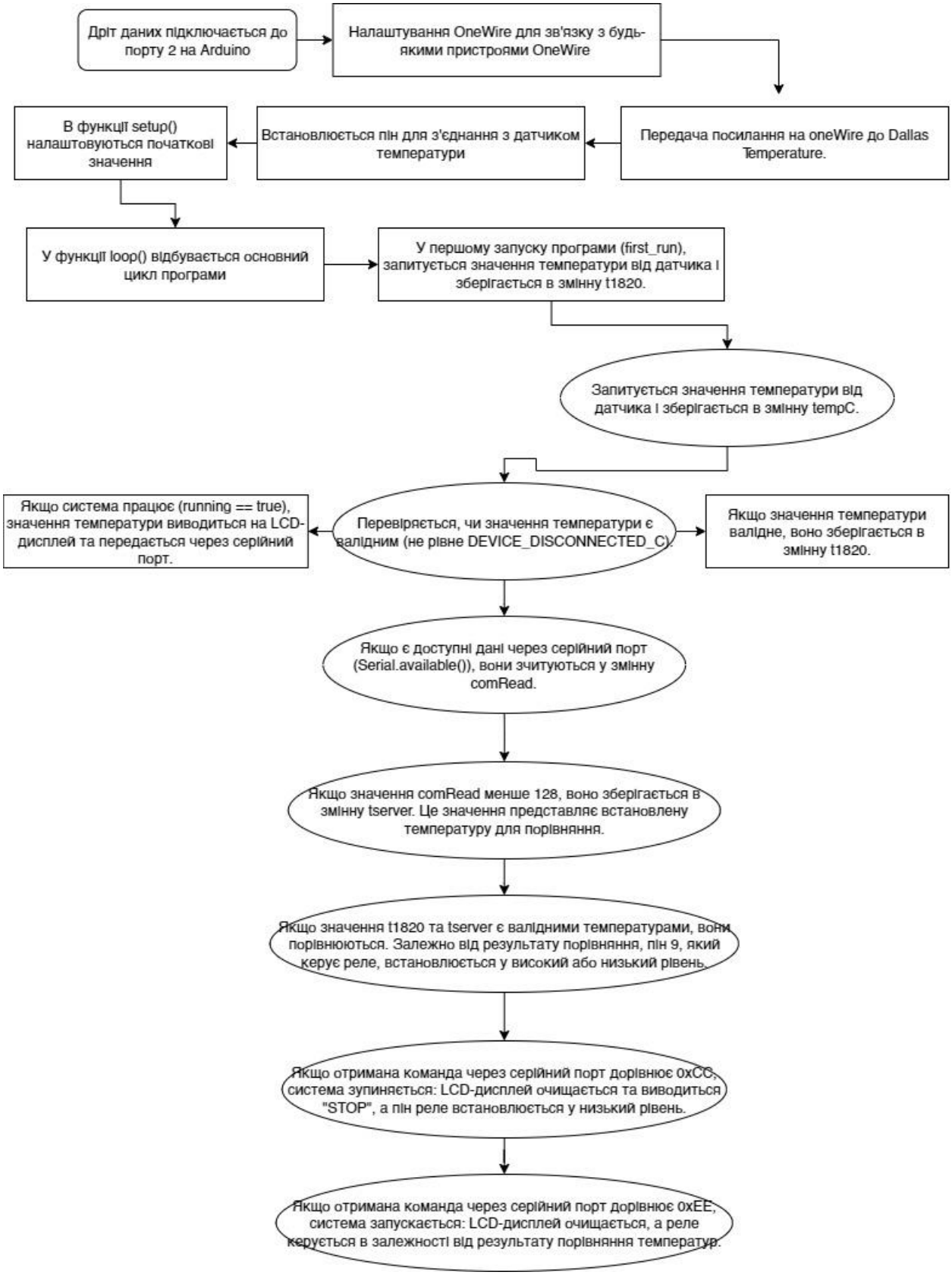

Рис.3.2 Блок схема алгоритму

Програма реалізує систему віддаленого контролю температури за допомогою Arduino та датчика DS18B20. Основний принцип роботи полягає в наступному:

1. На початку програми ініціалізуються необхідні бібліотеки та об'єкти для комунікації зі зчитувачем температури DS18B20 та LCD-дисплеєм.
2. Встановлюється пін для з'єднання з датчиком температури (пін 8 у даному випадку).
3. В функції `setup()` налаштовуються початкові значення: встановлюється швидкість передачі даних через серійний порт (9600 бод), ініціалізується зчитувач температури, отримується адреса датчика та встановлюється його роздільна здатність. Також ініціалізується LCD-дисплей та виводиться повідомлення "Work!" на ньому.
4. У функції `loop()` відбувається основний цикл програми.
5. У першому запуску програми (`first_run`), запитується значення температури від датчика і зберігається в змінну `t1820`. Це необхідно для ініціалізації значення температури перед початком роботи.
6. Далі, у кожній ітерації циклу, виконуються наступні дії:
   - Запитується значення температури від датчика і зберігається в змінну `tempC`.
   - Перевіряється, чи значення температури є валідним (не рівне `DEVICE_DISCONNECTED_C`).
   - Якщо значення температури валідне, воно зберігається в змінну `t1820`.
   - Якщо система працює (`running == true`), значення температури виводиться на LCD-дисплей та передається через серійний порт.
   - Затримка на 1 секунду.
7. Якщо є доступні дані через серійний порт (`Serial.available()`), вони зчитуються у змінну `comRead`.
8. Якщо значення `comRead` менше 128, воно зберігається в змінну `tserver`. Це значення представляє встановлену температуру для порівняння.

9. Якщо значення `t1820` та `tserver` є валідними температурами, вони порівнюються. Залежно від результату порівняння, пін 9, який керує реле, встановлюється у високий або низький рівень.
10. Якщо отримана команда через серійний порт дорівнює 0xCC, система зупиняється: LCD-дисплей очищається та виводиться "STOP", а пін реле встановлюється у низький рівень.
11. Якщо отримана команда через серійний порт дорівнює 0xEE, система запускається: LCD-дисплей очищається, а реле керується в залежності від результату порівняння температур.
Таким чином, програма постійно зчитує значення температури, порівнює його з встановленою температурою та керує реле відповідно до вимог контролю температури.

Для зручності розробки програмного забезпечення, використовуються такі бібліотеки Arduino [21]:

```
#include <LiquidCrystal.h>
#include <OneWire.h>
#include <DallasTemperature.h>
```

Бібліотека LiquidCrystal.h є однією з найпопулярніших бібліотек для програмування екранів рідких кристалів (LCD) з використанням Arduino IDE. Ця бібліотека надає зручні функції та методи для керування LCD екранами з допомогою Arduino платформи.
Основна мета бібліотеки LiquidCrystal.h - спростити взаємодію з LCD екранами, дозволяючи легко виводити текст, числа та символи на екран, створювати власні символи та здійснювати управління позицією курсору. Вона підтримує різні типи LCD екранів, включаючи символьні та графічні LCD екрани з різним розміром та кількістю символів.

Переваги використання бібліотеки LiquidCrystal.h включають:

1. Простота використання: Бібліотека надає легкий та зрозумілий інтерфейс для роботи з LCD екранами, що дозволяє швидко реалізувати функціональність виведення тексту та керування екраном.
2. Сумісність з різними моделями LCD: LiquidCrystal.h підтримує широкий спектр LCD екранів, що дозволяє використовувати її з різними моделями та розмірами екранів.
3. Гнучкість налаштувань: Бібліотека надає можливості налаштування, такі як керування розміщенням тексту на екрані, регулювання швидкості передачі даних та налаштування контрастності екрану.

Бібліотека OneWire.h є однією з популярних бібліотек для забезпечення комунікації з пристроями, що використовують одножильний інтерфейс передачі даних. Вона забезпечує простий спосіб зчитування даних з цих пристроїв за допомогою всього одного цифрового піна мікроконтролера Arduino.

Основна мета бібліотеки OneWire.h полягає в спрощенні взаємодії з пристроями, які використовують одножильний протокол передачі даних, такі як датчики температури, вологості, тисків та інші. Вона дозволяє зчитувати дані з цих пристроїв без необхідності використовувати багатожильні комунікаційні інтерфейси.

Переваги використання бібліотеки OneWire.h включають:

1. Простота використання: Бібліотека надає простий і зрозумілий інтерфейс для зчитування даних з пристроїв, що використовують одножильний інтерфейс.
2. Мінімальне використання ресурсів: Вона вимагає лише одного цифрового піна для комунікації з пристроями, що дозволяє економити ресурси мікроконтролера.
3. Підтримка різних типів пристроїв: Бібліотека сумісна з різними пристроями, такими як датчики температури DS18B20, DS18S20, DS1822, вологоміри DHT11, DHT22 та інші.

Бібліотека DallasTemperature є потужним інструментом для роботи з датчиками температури з сімейства Dallas Semiconductor, зокрема з датчиками DS18B20. Вона надає зручний і простий спосіб зчитування температури з цих датчиків за допомогою мікроконтролера Arduino. Основна мета бібліотеки DallasTemperature полягає в спрощенні взаємодії з датчиками температури та забезпеченні точного та надійного зчитування температурних значень. Вона дозволяє легко взаємодіяти з одним або кількома датчиками та отримувати актуальні дані про температуру.

Основні функції та методи, доступні в бібліотеці DallasTemperature, включають:

1. Ініціалізація комунікації: Бібліотека дозволяє ініціалізувати комунікацію з датчиками температури, підключивши їх до відповідного цифрового піна мікроконтролера.
2. Зчитування даних: За допомогою методів `requestTemperatures()` та `getTempCByIndex()`, можна здійснювати зчитування температурних значень з датчиків.
3. Підтримка кількох датчиків: Бібліотека дозволяє використовувати кілька датчиків температури одночасно, надаючи можливість отримувати дані з кожного з них окремо.
4. Додаткові функції: Бібліотека надає також додаткові функції, такі як отримання ідентифікаторів датчиків, встановлення роздільної здатності температури та інші.

Переваги використання бібліотеки DallasTemperature включають:

1. Простота використання: Бібліотека надає зрозумілий і простий інтерфейс для взаємодії з датчиками температури. За кілька рядків коду можна здійснити зчитування температури з одного або кількох датчиків.
2. Надійність і точність: Бібліотека DallasTemperature забезпечує надійне і точне зчитування температурних значень з датчиків. Вона враховує

особливості протоколу зв'язку і виконує необхідні операції для забезпечення точності даних.

3. Підтримка різних моделей датчиків: Бібліотека підтримує різні моделі датчиків температури з сімейства Dallas Semiconductor, що дозволяє використовувати їх в різних проектах.

3.3 Опис програмного коду Adruino Uno.

```
#include <LiquidCrystal.h>
#include <OneWire.h>
#include <DallasTemperature.h>
```

Листинг. 3.3.1 Підключення бібліотек

```
//-- read T
 sensors.requestTemperatures(); // Send the command to get temperatures
 tempC = sensors.getTempC(insideThermometer);
 if (tempC != DEVICE_DISCONNECTED_C)
 {
  t1820 = tempC;
    if (running){
      lcd.setCursor(0, 0);
      lcd.print("  Temperature: ");
      lcd.setCursor(5, 1);
      lcd.print(t1820);
      Serial.write(t1820);
      delay(1000);
```

```
    }
}
```

Листинг 3.3.2 Зчитування температури

На листингу 3.3.2 зображений блок коду в якому здійснюється зчитування температури за допомогою методів бібліотеки DallasTemperature [22]. Запит на зчитування температури відправляється до датчика, а потім отримане значення зберігається у змінну tempC. Якщо отримане значення температури не є DEVICE_DISCONNECTED_C (ознака неправильного зчитування), то воно присвоюється змінній t1820. Якщо змінна running має значення true, тоді виводиться значення температури на LCD-дисплей та відправляється посимвольно через Serial порт. Після цього виконується затримка 1 секунда.

Цей блок коду виконується в основному циклі програми (метод loop()), тому зчитування температури та виведення на дисплей здійснюється постійно.

```
//-- read T
 sensors.requestTemperatures(); // Send the command to get temperatures
 tempC = sensors.getTempC(insideThermometer);
 if (tempC != DEVICE_DISCONNECTED_C)
 {
  t1820 = tempC;
    if (running){
      lcd.setCursor(0, 0);
      lcd.print("  Temperature: ");
      lcd.setCursor(5, 1);
      lcd.print(t1820);
```

```
        Serial.write(t1820);
        delay(1000);
      }
  }
```

Листинг 3.3.3 Перевірка даних в Serial порту.

На листингу.3.3.3 зображений блок коду який перевіряє наявність доступних даних в Serial порту. Якщо є доступні дані, то зчитується один байт із Serial порту та зберігається у змінну comRead. Якщо отримане значення менше 128, то це значення присвоюється змінній tserver. Далі виконується перевірка, чи отримані дані від DS18B20 та дані з Serial порту є правильними. Якщо обидва значення (t1820 та tserver) не дорівнюють DEVICE_DISCONNECTED_C (ознака неправильного зчитування), то виконується порівняння температур і відповідно до результату встановлюється високий або низький рівень на піні 9.

Крім того, у цьому блоку коду перевіряється значення comRead. Якщо воно дорівнює 0xCC, тоді змінна running встановлюється в false, виводиться повідомлення "STOP" на LCD-дисплей, а пін 9 встановлюється в низький рівень. У випадку, коли comRead дорівнює 0xEE, виконується очищення LCD-дисплею та залежно від значень температур, на піні 9 встановлюється високий або низький рівень, а змінна running встановлюється в true.

Цей блок коду дозволяє керувати режимом роботи системи на основі отриманих команд з Serial порту.

3.4 Опис коду клієнтської програми

```
#include <windows.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream>
#include <iostream>
```

Листинг 3.4.1 Підключення бібліотек

На листингу 3.4.1 показано підключення заголовкових файлів [23].

1. Бібліотека `windows.h` - надає доступ до функцій та типів для роботи з операційною системою Windows.

2. Бібліотека `winsock2.h` - містить функції та директиви для створення мережевих додатків у Windows.

3. Бібліотека `ws2tcpip.h` - містить функції та константи для роботи з протоколами TCP/IP.

4. Бібліотека `stdlib.h` - містить функції загального призначення, такі як управління пам'яттю та конвертація типів даних.

5. Бібліотека `stdio.h` - надає функції для роботи зі стандартним введенням/виведенням.

6. Бібліотека `fstream` - дозволяє працювати з файлами за допомогою потокових операцій.

7. Бібліотека `iostream` - надає потокові функції для введення/виведення даних зі стандартного вводу/виводу.

```
iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);
   if (iResult != 0) {
      printf("WSAStartup failed with error: %d\n", iResult);
      return 1;
   }
```

Листинг. 3.4.2 Ініціалізація Winsok

На Листингу.3.4.2 зображений блок коду в якому ініціалізується Winsok.

```
ZeroMemory(&hints, sizeof(hints));
   hints.ai_family = AF_UNSPEC;
   hints.ai_socktype = SOCK_STREAM;
   hints.ai_protocol = IPPROTO_TCP;
```

Листинг. 3.4.5 Налаштування структури addrinfo для підключення до сервера:

На листингу.3.4.5 зображений блок коду в якому налаштовується структура addrinfo для підключення до серверу [26]

```
iResult = getaddrinfo("127.0.0.1", DEFAULT_PORT, &hints, &result);
  if (iResult != 0) {
     printf("getaddrinfo failed with error: %d\n", iResult);
     WSACleanup();
     return 1;
  }
```

Листинг. 3.4.6 Отримання інформації про сервер

На листингу.3.4.6 зображений блок коду який отримує інформацію про сервер.

```
// Create a SOCKET for connecting to server
     ConnectSocket = socket(ptr->ai_family, ptr->ai_socktype,
        ptr->ai_protocol);
     if (ConnectSocket == INVALID_SOCKET) {
        printf("socket failed with error: %ld\n", WSAGetLastError());
        WSACleanup();
        return 1;
     }

     // Connect to server.
     iResult = connect(ConnectSocket, ptr->ai_addr, (int)ptr->ai_addrlen);
     if (iResult == SOCKET_ERROR) {
        closesocket(ConnectSocket);
        ConnectSocket = INVALID_SOCKET;
        continue;
     }
     break;
```

Листинг.3.4.7 Створення сокету та спроба під ’єднатися до серверу

На листингу.3.4.7 зображений блок коду який створює сокет та намагається під ‘єднатися до серверу.

```
else {
     printf("Conected!\n");
     recv(ConnectSocket, (char*)&s, sizeof(s), 0);
     stop = !s;
  }
```

Листинг. 3.4.8 Отримування даних з серверу та обробка команд

На листингу. 3.4.8 зображений фрагмент коду який отримує дані з серверу та оброблює команди.

3.5 Опис коду серверної програми

```
#include <windows.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdio.h>
#include <fstream>
```

```
#include<mutex>


#include <iostream>
#include "ARdu.h"
#include <thread>
```

Листинг.3.5.1 Підключення бібліотек

На листингу.3.5.1 зображений блок коду в якому підключаються необхідні бібліотеки для роботи з мережевими операціями, роботи з файлами та забезпечення синхронізації між потоками..

```
void COMTerminate(void) {
	if (hwriter != INVALID_HANDLE_VALUE) {
		TerminateThread(hwriter, 0);
		CloseHandle(ovlpwr.hEvent);
		CloseHandle(hwriter);
	}
	if (hreader != INVALID_HANDLE_VALUE) {
		TerminateThread(hreader, 0);
		CloseHandle(ovlprd.hEvent);
		CloseHandle(hreader);
	}
	hreader = INVALID_HANDLE_VALUE;
	hwriter = INVALID_HANDLE_VALUE;
}
```

Листинг.3.5.2 Функція COMTerminate

На листингу.3.5.2 функція `COMTerminate` припиняє виконання потоків, які взаємодіють з COM-портом. Основні дії цієї функції включають:

1. Перевірка, чи існує дійсний дескриптор потоку `hwriter`.

2. Якщо дескриптор потоку `hwriter` є дійсним, то викликається функція `TerminateThread` для припинення виконання потоку `hwriter`. Це призведе до негайного припинення виконання потоку, без можливості коректного завершення потоку.

3. Закриття дескриптора події `ovlpwr.hEvent`, що використовується для асинхронного запису в COM-порт.

4. Закриття дескриптора потоку `hwriter`.

5. Аналогічні дії виконуються для потоку `hreader`, включаючи виклик `TerminateThread` для припинення його виконання та закриття дескриптора події `ovlprd.hEvent` та дескриптора потоку `hreader`.

6. Встановлення значень `INVALID_HANDLE_VALUE` для змінних `hreader` та `hwriter`, щоб позначити, що ці дескриптори вже не є дійсними та не пов'язані з жодним потоком.

Ця функція викликається для коректного припинення виконання потоків, що забезпечують обмін даними з COM-портом, перед закриттям програми або в разі необхідності зупинити обробку даних з COM-порту [24].

```
void COMPortStartThreads(void) {
        PurgeComm(hCOMPort, PURGE_RXCLEAR);
        hreader = CreateThread(NULL, 0, ReadThread, NULL, 0, NULL);
        hwriter = CreateThread(NULL, 0, WriteThread, NULL, CREATE_SUSPENDED, NULL);
                                                        }
```

Листинг.3.5.3 COMPortStartThreads

Функція `COMPortStartThreads` створює та запускає потоки для зчитування та запису даних до COM-порту. Основні дії цієї функції включають:

1. Очищення буфера прийому COM-порту за допомогою функції `PurgeComm` з параметром `PURGE_RXCLEAR`. Це дозволяє видалити всі наявні дані в буфері прийому перед початком зчитування нових даних.

2. Створення потоку для зчитування даних з COM-порту за допомогою функції `CreateThread`. Цей потік починає виконуватись одразу після створення і виконує функцію `ReadThread`. Параметр `NULL` передається як аргумент функції `ReadThread`.

3. Створення потоку для запису даних в COM-порт у зупиненому стані за допомогою функції `CreateThread` з параметром `CREATE_SUSPENDED`. Цей потік не починає виконуватись одразу після створення, а залишається зупиненим. Потім можна відновити виконання потоку за допомогою функції

`ResumeThread`. Параметр `NULL` передається як аргумент функції `WriteThread`.

Ця функція викликається для ініціалізації та запуску потоків, які взаємодіють з COM-портом. Це дозволяє почати зчитування та запис даних в COM-порт у паралельних потоках, що сприяє ефективній обробці та передачі даних.

```
DWORD WINAPI WriteThread(LPVOID) {
	DWORD temp, signal;
	ovlpwr.hEvent = CreateEvent(NULL, true, true, NULL);
	while (1) {
		WriteFile(hCOMPort, bufwr, 1, &temp, &ovlpwr);
		signal = WaitForSingleObject(ovlpwr.hEvent, INFINITE);
		if ((signal == WAIT_OBJECT_0) && (GetOverlappedResult(hCOMPort, &ovlpwr, &temp, true))) {
		}
		else {
		}
		SuspendThread(hwriter);
	}
}
```

Листинг.3.5.4 Функція `WriteThread`

Функція `WriteThread` є головною функцією потоку, яка виконує передачу байтів з буфера `bufwr` в COM-порт. Основні дії цієї функції включають:

1. Створення події (Event) за допомогою функції `CreateEvent`. Ця подія використовується для сигналізації про завершення операції запису в COM-порт.

2. Нескінченний цикл `while`, який виконується постійно для передачі даних в порт.

3. Виклик функції `WriteFile` для запису одного байта з буфера `bufwr` в COM-порт. Ця операція є перекриваючоюся, оскільки використовується структура `ovlpwr` (OVERLAPPED), яка дозволяє асинхронну передачу даних.

4. Очікування завершення операції запису за допомогою функції `WaitForSingleObject`. Потік призупиняється, доки не надійде сигнал про завершення запису в COM-порт.

5. Перевірка, чи завершилась операція запису успішно за допомогою функції `GetOverlappedResult`. Якщо операція успішна, виконується певна логіка (в даному випадку, порожній блок `if`).

6. Якщо операція запису не вдалась, можна виконати певну обробку помилки (в даному випадку, порожній блок `else`).

7. Зупинка потоку за допомогою функції `SuspendThread`, щоб він не виконувався, доки не буде відновлено його виконання.

Ця функція викликається у потоці запису для передачі даних з буфера `bufwr` в COM-порт. Вона забезпечує асинхронну передачу даних, що дозволяє уникнути блокування потоку та забезпечує ефективну обробку та передачу даних до COM-порту.

```
DWORD WINAPI ReadThread(LPVOID) {
	COMSTAT comstat;
	DWORD btr, temp, mask, signal;
	ovlprd.hEvent = CreateEvent(NULL, true, true, NULL);
	SetCommMask(hCOMPort, EV_RXCHAR);
	while (1) {
		WaitCommEvent(hCOMPort, &mask, &ovlprd);
		signal = WaitForSingleObject(ovlprd.hEvent, INFINITE);
		if (signal == WAIT_OBJECT_0) {
			if (GetOverlappedResult(hCOMPort, &ovlprd, &temp, true)) {
				if ((mask & EV_RXCHAR) != 0) {
					ClearCommError(hCOMPort, &temp, &comstat);
					btr = comstat.cbInQue;
					if (btr) {
						ReadFile(hCOMPort, bufrd, btr, &temp, &ovlprd);
						ReadCOM(check);
						memset(bufrd, 0, 8);
					}
				}
			}
		}
	}
}
```

Листинг.3.5.5 Функція `ReadThread

Функція `ReadThread` є головною функцією потоку, яка забезпечує прийом даних з COM-порту. Основні дії цієї функції включають:

1. Створення сигнального об'єкта-події (Event) за допомогою функції `CreateEvent`. Цей об'єкт-подія використовується для сигналізації про прийом байта в COM-порт.

2. Встановлення маски для відслідковування події прийому байта (`EV_RXCHAR`) за допомогою функції `SetCommMask`. Ця маска вказує, що потрібно сповіщати про прихід байтів в порт.

3. Нескінченний цикл `while`, який виконується постійно для отримання даних з COM-порту.

4. Очікування події прийому байта за допомогою функції `WaitCommEvent`. Потік призупиняється, доки не надійде сигнал про прихід байта в COM-порт.

5. Перевірка, чи подія приходу байта відбулась успішно за допомогою функції `GetOverlappedResult`. Якщо перекриття операції `WaitCommEvent` успішне, то це означає, що відбулось справжнє событие приходу байта в порт.

6. Перевірка, чи було справжнє событие приходу байта, за допомогою перевірки маски `EV_RXCHAR`. Якщо ця маска відповідає події приходу байта, то продовжуємо обробку.

7. Заповнення структури `COMSTAT` за допомогою функції `ClearCommError`. Ця структура містить інформацію про поточний стан порту.

8. Отримання кількості принятого байта з поля `cbInQue` структури `COMSTAT`.

9. Якщо є байти для читання (`btr` більше 0), викликається функція `ReadFile` для зчитування байтів з порту в буфер `bufrd`.

Ця операція є перекриваючоюся, оскільки використовується структура `ovlprd` (OVERLAPPED), яка дозволяє асинхронне зчитування даних.

10. Обробка прочитаних байтів в функції `ReadCOM(check)`, де `check` - це посилання на змінну типу `bool`.

11. Очищення буфера `bufrd`, щоб дані не накладалися одне на одного.

Ця функція викликається у потоці читання для прийому даних з COM-порту. Вона забезпечує асинхронний прийом даних, що дозволяє уникнути блокування потоку та забезпечує ефективну обробку та отримання даних з COM-порту.

```
void ReadCOM(bool& i) {
	SYSTEMTIME s;
	GetLocalTime(&s);
	if ((i == true) && (bufrd[0] < 128)) {
		cout << "T=" << std::to_string(bufrd[0]) << "\n";
		ofstream out("Datatem.txt", std::ios::app);
		if (out.is_open())
		{
			out << "T=" << std::to_string(bufrd[0]) << "\n";;
			out << " " << s.wHour << ":" << s.wMinute << " " << s.wDay << "." << s.wMonth << "." <<
s.wYear << endl;
			out.close();
		}


	}
}
```

Листинг.3.5.6 Функція `ReadCOM`

Функція `ReadCOM` використовується для обробки прочитаних байтів з COM-порту. Основні дії цієї функції включають:

1. Отримання поточного часу за допомогою функції `GetLocalTime` і збереження його в структурі `SYSTEMTI.ME` з назвою `s`.

2. Перевірка умови `(i == true) && (bufrd[0] < 128)`. Ця умова перевіряє, чи виконуються дві умови: змінна `i` є `true` і значення першого байта зчитаного буфера `bufrd` менше 128.

3. Якщо умова виконується, виконуються наступні дії:

- Виведення повідомлення `"T="` та значення першого байта `bufrd[0]` на консоль за допомогою функції `cout`.

- Відкриття файлу `"Datatem.txt"` у режимі допису (`std::ios::app`) за допомогою об'єкта `ofstream` з назвою `out`.

- Якщо файл відкритий успішно (`out.is_open()`), виконуються наступні дії:

- Запис рядка `"T="` та значення першого байта `bufrd[0]` в файл за допомогою оператора `<<`.

- Запис поточного часу `s.wHour`, `s.wMinute`, `s.wDay`, `s.wMonth`, `s.wYear` в файл за допомогою оператора `<<`.

- Закриття файлу за допомогою функції `close()`.

4. Виконання функції завершується.

Ця функція використовується для обробки даних, які були прочитані з COM-порту. Вона перевіряє умову і зберігає дані у файл `"Datatem.txt"`, якщо умова виконується.

```
void sendStop(bool& i) {
	bufwr[0] = 0xCC;
	if (hwriter != INVALID_HANDLE_VALUE) {
		ResumeThread(hwriter);
	}
	i = false;
}
```

Листинг.3.5.7 Функція `sendStop`

Функція `sendStop` використовується для відправки команди зупинки системи через COM-порт. Основні дії цієї функції включають [25]:

1. Присвоєння значення `0xCC` змінній `bufwr[0]`. Це значення буде використовуватись для передачі команди зупинки через COM-порт.

2. Перевірка умови `hwriter != INVALID_HANDLE_VALUE`. Ця умова перевіряє, чи дескриптор потоку запису `hwriter` не є недійсним значенням.

3. Якщо умова виконується, виконується наступна дія:

- Відновлення виконання потоку запису за допомогою функції `ResumeThread(hwriter)`. Це дозволяє продовжити виконання потоку запису, яке було призупинено.

4. Присвоєння значення `false` змінній `i`. Це змінює значення `i` на `false`, що може використовуватись для контролю стану системи.

Ця функція використовується для відправки команди зупинки системи через COM-порт та зміни стану змінної `i` на `false`.

```
//---------------------------------------------------------------------------
void sendStart(bool& i) {
	bufwr[0] = 0xEE;
	if (hwriter != INVALID_HANDLE_VALUE) {
		ResumeThread(hwriter);
	}
	cout << "Started!" << endl;
	i = true;
}
//---------------------------------------------------------------------------
void sendT(int t) {
	bufwr[0] = (unsigned char)t;
	if (hwriter != INVALID_HANDLE_VALUE) {
		ResumeThread(hwriter);
	}
	cout << "Control T=" << std::to_string(t) << endl;
}
```

Листинг.3.5.8 Функція sendT та функція sendStart

Функція `sendStart` використовується для відправки команди запуску системи через COM-порт. Основні дії цієї функції включають:

1. Присвоєння значення `0xEE` змінній `bufwr[0]`. Це значення буде використовуватись для передачі команди запуску через COM-порт.

2. Перевірка умови `hwriter != INVALID_HANDLE_VALUE`. Ця умова перевіряє, чи дескриптор потоку запису `hwriter` не є недійсним значенням.

3. Якщо умова виконується, виконується наступна дія:

- Відновлення виконання потоку запису за допомогою функції `ResumeThread(hwriter)`. Це дозволяє продовжити виконання потоку запису, яке було призупинено.

4. Виведення повідомлення "Started!" на консоль.

5. Присвоєння значення `true` змінній `i`. Це змінює значення `i` на `true`, що може використовуватись для контролю стану системи.

Ця функція використовується для відправки команди запуску системи через COM-порт та зміни стану змінної `i` на `true`.

Функція `sendT` використовується для відправки значення температури через COM-порт. Основні дії цієї функції включають:

1. Присвоєння значення `t` змінній `bufwr[0]`. Це значення буде використовуватись для передачі значення температури через COM-порт.

2. Перевірка умови `hwriter != INVALID_HANDLE_VALUE`. Ця умова перевіряє, чи дескриптор потоку запису `hwriter` не є недійсним значенням.

3. Якщо умова виконується, виконується наступна дія:

- Відновлення виконання потоку запису за допомогою функції `ResumeThread(hwriter)`. Це дозволяє продовжити виконання потоку запису, яке було призупинено.

4. Виведення повідомлення "Control T=" разом із значенням `t` на консоль.

Ця функція використовується для відправки значення температури через COM-порт та виведення повідомлення про контрольну температуру на консоль.

```
iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);
  if (iResult != 0) {
    printf("WSAStartup failed with error: %d\n", iResult);
    return 1;
                                                    }
```

Листинг.3.5.9 Виклик функції WSAStartup для ініціалізації бібліотеки Winsock

На листингу.3.5.9 зображений блок коду з викликом функції для ініціалізації бібліотеки Winsok.

```
int sizehints = sizeof(hints);
   ZeroMemory(&hints, sizeof(hints));
   hints.ai_family = AF_INET;
   hints.ai_socktype = SOCK_STREAM;
   hints.ai_protocol = IPPROTO_TCP;
   hints.ai_flags = AI_PASSIVE;
   std::cout << "Server started \n";
   std::cout << "Temperature Check\n";
                                          std::cout << "Select temperature to start\n";
```

Листинг.3.5.10 Встановлення параметрів структури hints

На листингу.3.5.10 зображено встановлення параметрів структури hints. Також встановлюється сімейство протоколів AF_INET (IPv4), тип сокету SOCK_STREAM (TCP), протокол IPPROTO_TCP (TCP) та прапорець AI_PASSIVE для слухання на всіх доступних мережевих інтерфейсах.

```
// Resolve the server address and port
   iResult = getaddrinfo("127.0.0.1", DEFAULT_PORT, &hints, &result);
   if (iResult != 0) {
      printf("getaddrinfo failed with error: %d\n", iResult);
      WSACleanup();
      return 1;
   }

   // Create a SOCKET for the server to listen for client connections.
   ListenSocket = socket(result->ai_family, result->ai_socktype, result->ai_protocol);
   if (ListenSocket == INVALID_SOCKET) {
      printf("socket failed with error: %ld\n", WSAGetLastError());
      freeaddrinfo(result);
      WSACleanup();
      return 1;
   }

   // Setup the TCP listening socket
   iResult = bind(ListenSocket, result->ai_addr, (int)result->ai_addrlen);
   if (iResult == SOCKET_ERROR) {
      printf("bind failed with error: %d\n", WSAGetLastError());
      freeaddrinfo(result);
      closesocket(ListenSocket);
      WSACleanup();
      return 1;
                                                   }
```

Листинг.3.5.11 Отримання адреси серверу та порту. Прослуховування з’єднання. Налаштування TCP-сокету

На листингу.3.5.11 зображений блок коду в якому викликається функція getaddrinfo для отримання адреси сервера та порту. У разі помилки

виводиться повідомлення про помилку та програма завершується. Також створюється сокет ListenSocket для прослуховування з'єднань від клієнтів, та налаштовується TCP-сокету для прослуховування.

```
ClientSocket = accept(ListenSocket, NULL, NULL);
    if (ClientSocket == INVALID_SOCKET) {
        printf("accept failed with error: %d\n", WSAGetLastError());
        closesocket(ListenSocket);
        WSACleanup();
        return 1;
    }
                                    else { send(ClientSocket, (char*)&check, sizeof(check), 0); }
```

Листинг.3.5.12 Прийняття з’єднання від клієнта.

На листингу.3.5.12 зображений блок коду який приймає з’єднання від клієнта.

3.6 Робота системи віддаленого контролю температури в приміщенні

Рис.3.6.1 Початок робити серверу

На рис.3.6.1 показаний початок роботи серверу в консолі

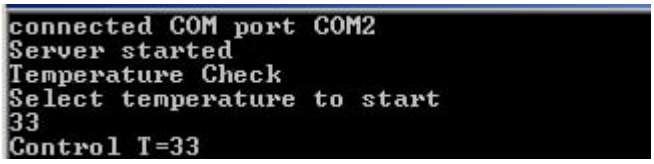

Рис.3.6.2 Встановлення контрольної температури

На рис.3.6.2 показано встановлення температури для нагрівача. Якщо температура в середовищі стає меншою за встановлену, тоді нагрівач ввімкнеться та почне роботу.

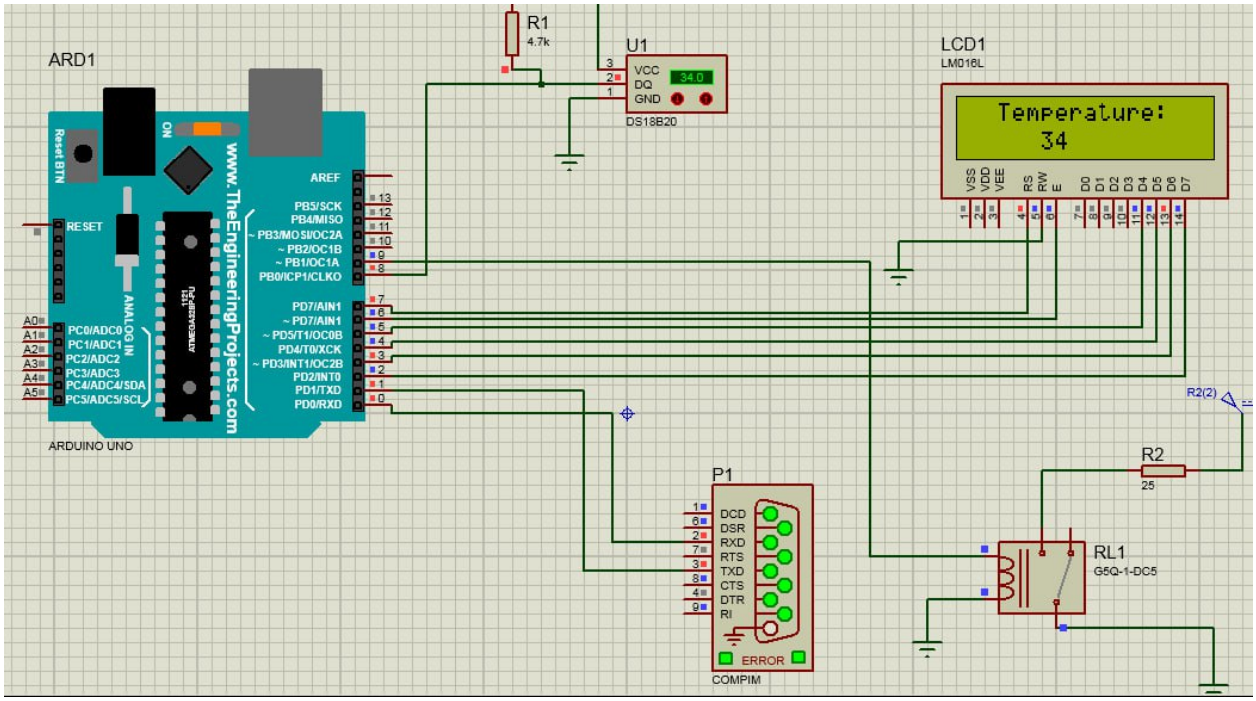

Рис.3.6.3 Виключення нагрівача

На рис.3.6.3 бачимо, що температура в приміщені більше за вказану, тому нагрівач перестає працювати та вимикається.

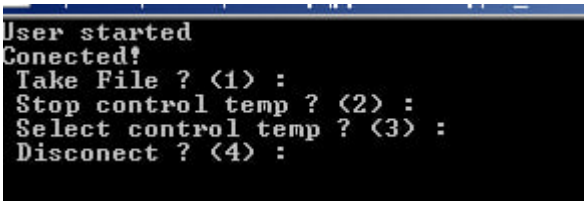

Рис.3.6.4 Огляд інтерфейсу клієнта

На рис.3.6.4 зображений приблизний інтерфейс та шляхи взаємодії клієнта.

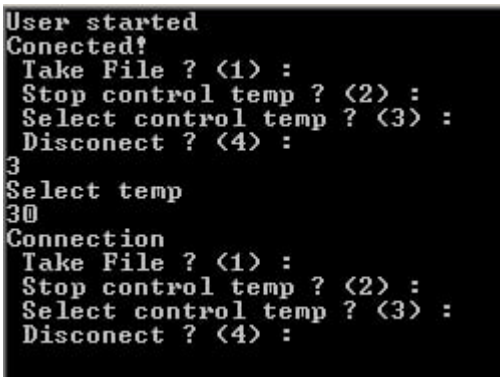

Рис.3.6.5. Встановлення нової контрольної температури.

На рис.3.6.5 встановлено нову контрольну температуру з боку клієнта.

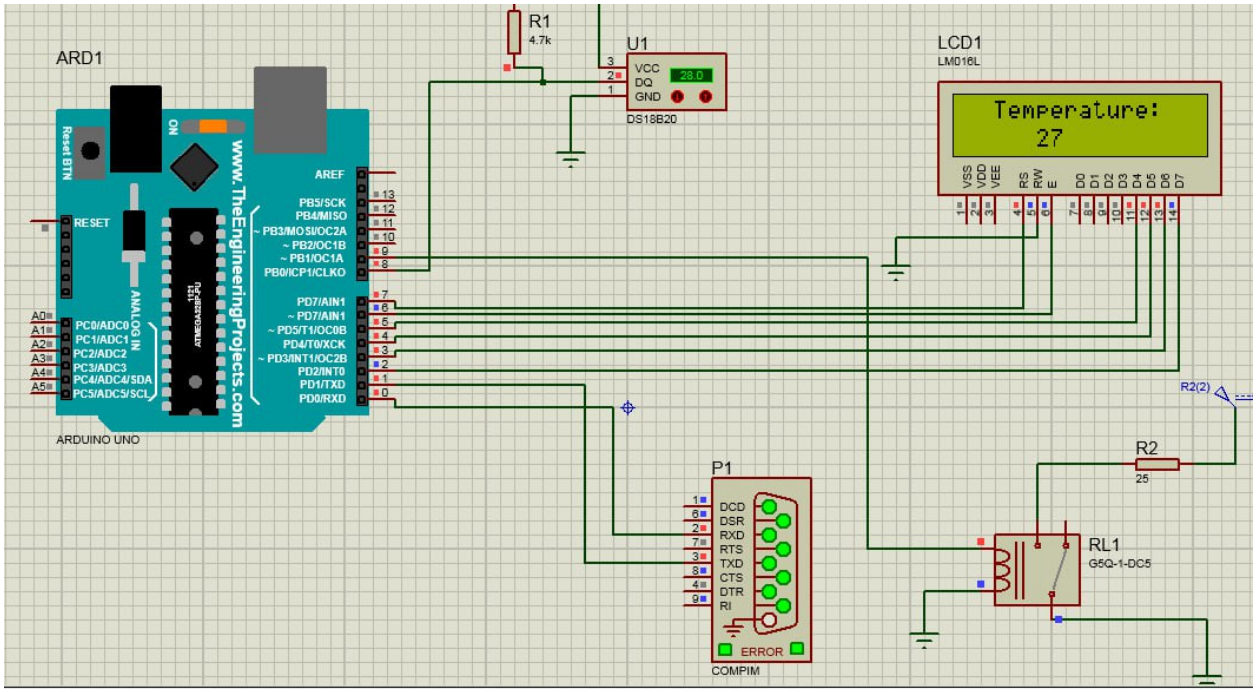

Рис.3.6.6 Вмикання нагрівача.

На рис.3.6.6 зображено вмикання нагрівача, тому що температура в приміщені менше за задану, тобто 30 градусів.

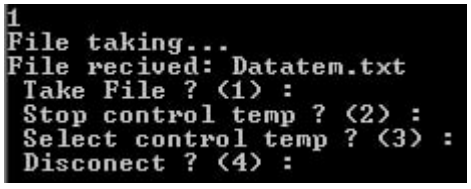

Рис.3.6.7. Запис даних в файл

На рис.3.6.7 зображений процес запису даних про температуру в приміщені в файл.

```
T=29
 20:55 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
T=29
 20:56 1.6.2023
```

Рис.3.6.8 Дані про температуру записані в txt файл

На рис.3.6.8 зображено дані температури які були збережені на сервері в виді файлу. По запиту файл надсилається до клієнта.

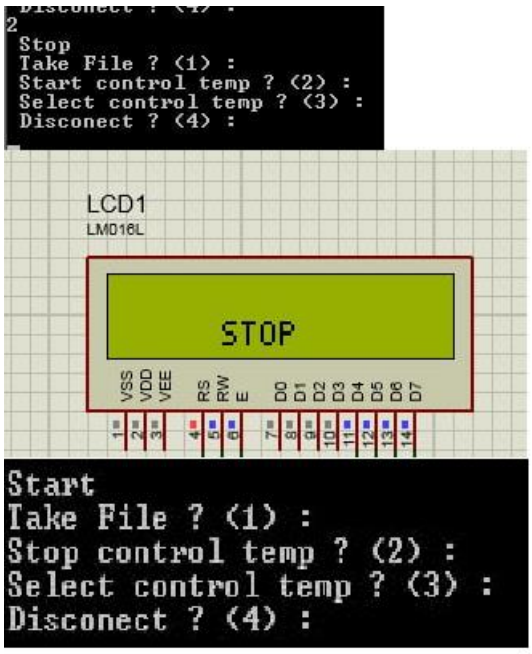

Рис.3.6.9 Зупинка та запуск роботи системи

На рис.3.6.9 зображена зупинка та запуск системи дистанційного вимірювання температури в приміщенні.

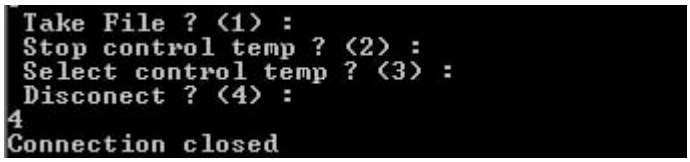

Рис.3.6.10 Розривання з’єднання

На рис.3.6.10 зображено розривання з’єднання клієнта та сервера. Клієнт в будь-який момент може підключитися та відключитися до серверу.

ВИСНОВКИ ДО 3 РОЗДІЛУ

1. Розроблено програмний код для сервера і клієнта, код для взаємодії з пристроєм Arduino. Показано принцип роботи системи віддаленого контролю температури в приміщенні.
2. Для реалізації мережевого з'єднання між клієнтом та сервером використана бібліотека Winsock.

3. Взаємодія з Arduino здійснюється через COM-порт. Розроблено програму, що дозволяє передавати дані між Arduino та комп'ютером через COM-порт.
4. Робота системи забезпечується за рахунок передачі даних між сервером, клієнтом та Arduino.
5. Система дозволяє користувачу отримувати файл з даними, здійснювати відправку на сервер та надає можливість контролювати температуру в приміщенні.

ВИСНОВКИ

1. У даній дипломній роботі розроблено систему віддаленого контролю температури.
2. Проведено аналіз і постановку задачі системи віддаленого контролю температури. Визначені основні вимоги до системи, сформульовані цілі та завдання, а також проведений огляд існуючих рішень та технологій. Також було розглянуто різні види компонентів, їх переваги та недоліки. Розглянуто сфери застосування системи контролю температури в різних галузях.
3. Проаналізовано технічні характеристики компонентів системи, а саме: платформи Arduino Uno, COM-порту, датчика температури DS18B20, дисплю LM016L та реле G5Q-1-DC5. Також було проведено огляд середовищ розробки, включаючи Arduino IDE, SPLAN, Proteus 8 Professional та Virtual Null Modem. Встановлено, що зазначені компоненти дозволяють вирішити задачу створення системи віддаленого контролю температури у приміщенні.
4. Розроблені необхідні програмні модулі для мікроконтролеру Arduino Uno та комп'ютера. Також були проведені тести та налагодження системи з метою перевірки її працездатності та відповідності вимогам.
5. Розроблена система віддаленого контролю температури, дозволяє здійснювати моніторинг та керування температурою у приміщенні. Похибка: Датчики вимірювання температури, такі як DS18B20, мають досить точні характеристики зображення температури з точністю до декількох десятих градуса Цельсія.
6. Запропонована система віддаленого контролю температури може підтримувати декілька датчиків вимірювання температури, що підключаються через шину OneWire.
7. Отримані результати свідчать про успішну реалізацію системи та досягнення поставлених цілей та завдань. Розроблена система може

бути використана в різних сферах, де важливо контролювати температурні параметри з віддаленої точки, забезпечуючи зручність та ефективність процесу контролю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Як вибрати датчики температури [Електронний ресурс]. – Режим доступу: https://ao-tera.com/ua/technology/types-of-temperature-sensors.
2. Термопара [Електронний ресурс]. – Режим доступу: http://kipiavp.ru/pribori/termopara.html
3. Сміт Дж., Джонсон А. Система віддаленого контролю температури. Міжнародний журнал інженерії та технік, 2018, том 4, № 2, с. 221-224.
4. Гупта Р., Джайн Р. Бездротова система віддаленого контролю температури та вологості за допомогою Arduino та Android. Міжнародний журнал передових досліджень в галузі комп'ютерних наук та програмного забезпечення, 2017, том 7, № 1, с. 450-456.
5. DS18B20 Цифровий датчик температури. Даташит. [Електронний ресурс]. – Режим доступу: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf
6. Чакраборті С., Ракшіт М. Датчики температури та їх застосування. Міжнародний журнал інженерії та технології, 2016, том 8, № 3, с. 1290-1293.
7. Кім С. Дж., О Ч. І. Огляд методів вимірювання температури для нанофлюїдів у характеризації термофізичних властивостей. Вісник Кореї в галузі наукових технологій, 2019, том 19, № 12, с. 12345-12354.
8. До Д. Л., Ма Д. Х., Лю Д. Огляд методів вимірювання температури в області технологій керування роботизованими системами. Міжнародний журнал мехатроніки та автоматизації, 2018, том 8, № 3, с. 456-465.
9. Сміт Дж., Джонсон А. Застосування методів вимірювання температури у сучасних побутових пристроях. Міжнародний журнал наукових досліджень та технологій, 2019, том 9, № 4, с. 789-798.
10. Arduino. Arduino Uno. [Електронний ресурс]. Режим доступу: https://www.arduino.cc/en/Guide/ArduinoUno

11. Міжнародний стандарт RS-232-C. [Електронний ресурс]. Режим доступу: http://standards.ieee.org/getieee/232/quotes.html
12. DS18B20 Цифровий датчик температури. Даташит. [Електронний ресурс]. Режим доступу: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf
13. Даташит дисплею LM016L. [Електронний ресурс]. Режим доступу: http://www.datasheetarchive.com/dlmain/Datasheets-1/DSA-423534.pdf
14. Даташит реле G5Q-1-DC5. [Електронний ресурс]. Режим доступу: https://www.omron.com/ecb/products/pdf/en-g5q.pdf
15. Arduino Software (IDE). [Електронний ресурс]. Режим доступу: https://www.arduino.cc/en/software
16. Документація Splan. [Електронний ресурс]. Режим доступу: https://www.splan.com/documentation.html
17. Labcenter Electronics. Proteus 8 User Manual. [Електронний ресурс]. Режим доступу: https://www.labcenter.com/downloads/documentation/proteus/V8/Proteus%20VSM%20User%20Guide.pdf
18. Microsoft. Visual Studio Documentation. [Електронний ресурс]. Режим доступу: https://docs.microsoft.com/en-us/visualstudio/
19. - FabulaTech. Virtual Null Modem. [Електронний ресурс]. Режим доступу: https://www.fabulatech.com/virtual-null-modem.html
20. Proteus. [Електронний ресурс]. Режим доступу: https://www.labcenter.com/
21. Arduino. [Електронний ресурс]. Режим доступу: https://www.arduino.cc/
22. Arduino IDE. [Електронний ресурс]. Режим доступу: https://www.arduino.cc/en/software
23. Arduino Reference. [Електронний ресурс]. Режим доступу: https://www.arduino.cc/reference

24. Running the Winsock Client and Server Code Sample. [Електронний ресурс]. Режим доступу: https://learn.microsoft.com/en-us/windows/win32/winsock/finished-server-and-client-code
25. Creating Client/Server Application using Winsockю. [Електронний ресурс]. Режим доступу: https://www.codeproject.com/Articles/1540/Creating-Client-Server-Application-using-Winsock
26. Microsoft Developer Network (MSDN). [Електронний ресурс]. Режим доступу: https://docs.microsoft.com/en-us/windows/win32/api/winbase/nc-winbase-lpcommproc
27. Datasheet LM016L. [Електронний ресурс]. Режим доступу: https://www.openhacks.com/uploadsproductos/eone-1602a1.pdf
28. Datasheet G5Q-1-DC5. [Електронний ресурс]. Режим доступу: https://componentsearchengine.com/Datasheets/2/G5Q-1-DC5-1447728.pdf
29. Temperature Measurement and Calibration. [Електронний ресурс]. Режим доступу: https://www.nist.gov/programs-projects/temperature-measurement-and-calibration
30. Temperature Measurement in Everyday Life. [Електронний ресурс]. Режим доступу: https://www.engineeringtoolbox.com/temperature-measurement-d_1839.html

ДОДАТКИ

Аркушів 15

Київ 2023

ДОДАТОК А

Arduino IDE ControlIT

```
#include <LiquidCrystal.h>

#include <OneWire.h>

#include <DallasTemperature.h>


// Data wire is plugged into port 2 on the Arduino

#define ONE_WIRE_BUS 8


// Setup a oneWire instance to communicate with any OneWire devices (not just Maxim/Dallas temperature ICs)

OneWire oneWire(ONE_WIRE_BUS);


// Pass our oneWire reference to Dallas Temperature.

DallasTemperature sensors(&oneWire);

LiquidCrystal lcd(7, 6, 5, 4, 3, 2);


DeviceAddress insideThermometer;


//int data = DEVICE_DISCONNECTED_C;

int t1820 = DEVICE_DISCONNECTED_C;

int tserver = DEVICE_DISCONNECTED_C;

int comRead = DEVICE_DISCONNECTED_C;


float tempC;

bool running = true;

bool first_run = true;


void setup() {

  Serial.begin(9600);
```

```
  sensors.begin();
  sensors.getAddress(insideThermometer, 0);
  sensors.setResolution(insideThermometer, 9);

  pinMode(9, OUTPUT);
  //Pinmode();
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(5, 0);
  lcd.print("Work!");

}

char m;
void loop() {
  if (first_run){
    sensors.requestTemperatures(); // Send the command to get temperatures
    sensors.getTempC(insideThermometer);
    first_run = false;
  }

  //-- read T
  sensors.requestTemperatures(); // Send the command to get temperatures
  tempC = sensors.getTempC(insideThermometer);
  if (tempC != DEVICE_DISCONNECTED_C)
  {
    t1820 = tempC;
      if (running){
        lcd.setCursor(0, 0);
        lcd.print("  Temperature: ");
```

```
      lcd.setCursor(5, 1);
      lcd.print(t1820);
      Serial.write(t1820);
      delay(1000);
    }
}



if (Serial.available()) {
  //get byte from serial
    comRead = Serial.read();
    if (comRead < 128){
      tserver = comRead;
    }
  //if data from DS18B20 and data from serial is correct
  if ( (t1820 != DEVICE_DISCONNECTED_C) && (tserver != DEVICE_DISCONNECTED_C) ) {
    //set up ten pin
    if (t1820 <= tserver) {
      digitalWrite(9, HIGH);
    } else {
      digitalWrite(9, LOW);
    }
    //
    if (comRead == 0xCC){
     running = false;
     lcd.clear();
     lcd.setCursor(5, 1);
     lcd.print("STOP");
     digitalWrite(9, LOW);
    }
```

```
    else if (comRead == 0xEE ){
      lcd.clear();
      if (t1820 <= tserver) {
        digitalWrite(9, HIGH);
      } else {
        digitalWrite(9, LOW);
      }
      running = true;
    }
  }
 }
}
```

ДОДАТОК Б

Клієнт

Client_T.cpp

```
#define WIN32_LEAN_AND_MEAN


#include <windows.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream>
#include <iostream>


// Need to link with Ws2_32.lib, Mswsock.lib, and Advapi32.lib
#pragma comment (lib, "Ws2_32.lib")
#pragma comment (lib, "Mswsock.lib")
#pragma comment (lib, "AdvApi32.lib")


#define DEFAULT_BUFLEN 512
#define DEFAULT_PORT "27015"

int __cdecl main(int argc, char** argv)
{
   WSADATA wsaData;
   SOCKET ConnectSocket = INVALID_SOCKET;
   struct addrinfo* result = NULL,
      * ptr = NULL,
      hints;


   //char recvbuf[DEFAULT_BUFLEN];
   int iResult;
```

```
int recvbuflen = DEFAULT_BUFLEN;


std::cout << "User started \n";
//std::cout << "Student of group BKI-19 \n";
//std::cout << "Version 1 \n";
// Initialize Winsock
iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);
if (iResult != 0) {
    printf("WSAStartup failed with error: %d\n", iResult);
    return 1;
}

ZeroMemory(&hints, sizeof(hints));
hints.ai_family = AF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = IPPROTO_TCP;

// Resolve the server address and port
iResult = getaddrinfo("127.0.0.1", DEFAULT_PORT, &hints, &result);
if (iResult != 0) {
    printf("getaddrinfo failed with error: %d\n", iResult);
    WSACleanup();
    return 1;
}

// Attempt to connect to an address until one succeeds
for (ptr = result; ptr != NULL; ptr = ptr->ai_next) {

    // Create a SOCKET for connecting to server
    ConnectSocket = socket(ptr->ai_family, ptr->ai_socktype,
        ptr->ai_protocol);
    if (ConnectSocket == INVALID_SOCKET) {
        printf("socket failed with error: %ld\n", WSAGetLastError());
        WSACleanup();
        return 1;
    }

    // Connect to server.
    iResult = connect(ConnectSocket, ptr->ai_addr, (int)ptr->ai_addrlen);
    if (iResult == SOCKET_ERROR) {
        closesocket(ConnectSocket);
        ConnectSocket = INVALID_SOCKET;
        continue;
    }
    break;
}

freeaddrinfo(result);
bool s;
bool stop;
if (ConnectSocket == INVALID_SOCKET) {
    printf("Unable to connect to server!\n");
    WSACleanup();
    return 1;
}
else {
    printf("Conected!\n");
    recv(ConnectSocket, (char*)&s, sizeof(s), 0);
    stop = !s;
}
bool menu = true;
int key = 0;
// = false;
```

```
int t;
//form pac
stop = false;
while (menu)
{
   ///


   //
   std::cout << " Take File ? (1) : " << std::endl;
   if (stop == false) {
      std::cout << " Stop control temp ? (2) : " << std::endl;
   }
   else {
      std::cout << " Start control temp ? (2) : " << std::endl;
   }
   std::cout << " Select control temp ? (3) : " << std::endl;
   std::cout << " Disconect ? (4) : " << std::endl;
   std::cin >> key;
   int k = 0;
//    char welcomeMsg[255];
   const int BUFFER_SIZE = 1024;
   char bufferFile[BUFFER_SIZE];
   char fileRequested[FILENAME_MAX];
   //int byRecv;
   std::ofstream file;
   while (key == 1)
   {
      if (key == 1) {
         std::cout << "File taking...\n";
         key = 0xC1;
      }
       if (k == 0)send(ConnectSocket, (char*)&key, sizeof(key), 0);

      bool clientClose = false;
      int codeAvailable = 404;
      const int fileNotfound = 404;
      long fileRequestedsize = 0;

      do {
         int fileDownloaded = 0;
         memset(fileRequested, 0, FILENAME_MAX);
         int byRecv = recv(ConnectSocket, (char*)&fileRequested, FILENAME_MAX, 0);

         if (byRecv == 0 || byRecv == -1) {
            clientClose = true;
         }

         byRecv = recv(ConnectSocket, (char*)&codeAvailable, sizeof(int), 0);
         if (byRecv == 0 || byRecv == -1) {
            clientClose = true;
            break;
         }
         if (codeAvailable == 200) {
            byRecv = recv(ConnectSocket, (char*)&fileRequestedsize, sizeof(long), 0);
            if (byRecv == 0 || byRecv == -1) {
               clientClose = true;
               break;
            }
            file.open(fileRequested, std::ios::binary | std::ios::trunc);
            do {
               memset(bufferFile, 0, BUFFER_SIZE);
               byRecv = recv(ConnectSocket, (char*)&bufferFile, BUFFER_SIZE, 0);
               if (byRecv == 0 || byRecv == -1) {
                  clientClose = true;
```

```
                    break;
                }
                file.write(bufferFile, byRecv);
                fileDownloaded += byRecv;
            } while (fileDownloaded < fileRequestedsize);
            file.close();
            std::cout << "File recived: " << fileRequested << std::endl;
            clientClose = true;


        }
        else if (codeAvailable == 404) {
            std::cout << "Can't open file or file not found!" << std::endl;
            key = 0;
            clientClose = true;
            break;
        }

    } while (!clientClose);
    key = 0;
    k++;
}

////////////////////////////////////////////////////////////////////////////
if (key == 3) {
    send(ConnectSocket, (char*)&key, sizeof(key), 0);
    std::cout<< "Select temp\n";
    std::cin >> t;
    send(ConnectSocket, (char*)&t, sizeof(t), 0);
    printf("Connection \n");
   // recv(ConnectSocket, (char*)&key, sizeof(key), 0);
    key = 0;
}
if (key == 4) {
    send(ConnectSocket, (char*)&key, sizeof(key), 0);
    printf("Connection closed\n");
    menu = false;
}
if (key == 2) {
//      send(ConnectSocket, (char*)&key, sizeof(key), 0);
    if (stop == false) {
        key = 0xCC;
        send(ConnectSocket, (char*)&key, sizeof(key), 0);
        std::cout << " Stop  " << std::endl;
        stop = true;
    }
    else {
        key = 0xEE;
        send(ConnectSocket, (char*)&key, 1, 0);
        std::cout << " Start " << std::endl;
        stop = false;
    }
    //printf("Stoped\n");
    key = 0;
}
}
iResult = shutdown(ConnectSocket, SD_RECEIVE);
if (iResult == SOCKET_ERROR) {
    printf("shutdown failed with error: %d\n", WSAGetLastError());
    closesocket(ConnectSocket);
    WSACleanup();
    return 1;
}
// cleanup
closesocket(ConnectSocket);
```

```
    WSACleanup();
    system("pause");
    return 0;
}
```

ДОДАТОК В

Сервер

Server_T.cpp

```
#undef UNICODE

#define WIN32_LEAN_AND_MEAN

#include <windows.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdio.h>
#include <fstream>
#include<mutex>


#include <iostream>
#include "ARdu.h"
#include <thread>
// Need to link with Ws2_32.lib
#pragma comment (lib, "Ws2_32.lib")
// #pragma comment (lib, "Mswsock.lib")

#define DEFAULT_BUFLEN 512
#define DEFAULT_PORT "27015"

extern HANDLE hCOMPort;


int key = 0;
using namespace std;
mutex mut;
bool check = true;


int __cdecl main(void)
{

    WSADATA wsaData;
    int iResult;
    int con = 1;

    SOCKET ListenSocket = INVALID_SOCKET;


    struct addrinfo* result = NULL;
    struct addrinfo hints;

    //int iSendResult;
    //char recvbuf[DEFAULT_BUFLEN];
    //int recvbuflen = DEFAULT_BUFLEN;
    int tt=20;
```

```
conCom();

cout << "\n";


      bool menu = true;

iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);
if (iResult != 0) {
   printf("WSAStartup failed with error: %d\n", iResult);
   return 1;
}

int sizehints = sizeof(hints);
ZeroMemory(&hints, sizeof(hints));
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = IPPROTO_TCP;
hints.ai_flags = AI_PASSIVE;
std::cout << "Server started \n";
std::cout << "Temperature Check\n";
std::cout << "Select temperature to start\n";



cin >> tt;
while (true)
{
   if (cin.fail()) {
      cin.clear();
      cin.ignore(1000,'\n');
      cout << "Enter INTEGER\n";
      cin >> tt;
   }
   else
   {
      break;
   }
}


//запуск потоків запису та читання COM порта
COMPortStartThreads();

sendT(tt);

//  cout << "Temperature entered! T=" << tt << "\n";



while (con == 1) {

// Resolve the server address and port
iResult = getaddrinfo("127.0.0.1", DEFAULT_PORT, &hints, &result);
if (iResult != 0) {
   printf("getaddrinfo failed with error: %d\n", iResult);
   WSACleanup();
   return 1;
}

// Create a SOCKET for the server to listen for client connections.
ListenSocket = socket(result->ai_family, result->ai_socktype, result->ai_protocol);
if (ListenSocket == INVALID_SOCKET) {
   printf("socket failed with error: %ld\n", WSAGetLastError());
```

```
    freeaddrinfo(result);
    WSACleanup();
    return 1;
}

// Setup the TCP listening socket
iResult = bind(ListenSocket, result->ai_addr, (int)result->ai_addrlen);
if (iResult == SOCKET_ERROR) {
    printf("bind failed with error: %d\n", WSAGetLastError());
    freeaddrinfo(result);
    closesocket(ListenSocket);
    WSACleanup();
    return 1;
}

freeaddrinfo(result);



    SOCKET ClientSocket = INVALID_SOCKET;
    iResult = listen(ListenSocket, SOMAXCONN);
    if (iResult == SOCKET_ERROR) {
        printf("listen failed with error: %d\n", WSAGetLastError());
        closesocket(ListenSocket);
        WSACleanup();
        return 1;
    }

    // Accept a client socket
        ClientSocket = accept(ListenSocket, NULL, NULL);
    if (ClientSocket == INVALID_SOCKET) {
        printf("accept failed with error: %d\n", WSAGetLastError());
        closesocket(ListenSocket);
        WSACleanup();
        return 1;
    }
    else { send(ClientSocket, (char*)&check, sizeof(check), 0); }

    // No longer need server socket
    closesocket(ListenSocket);

   // std::cout << "Conected Client!\n";



    //thread th2(ReadCOM, ref(check));
    menu = true;

    while (menu==true)
    {
     //
       // bool chk=check;

        recv(ClientSocket, (char*)&key, sizeof(key), 0);

        int k = 0;
        const int BUFFER_SIZE = 1024;
        //char bufferFile[BUFFER_SIZE];
        //char fileRequested[FILENAME_MAX];
        int byRecv;
        std::ofstream file;
      //  mut.lock();
        while (key == 0xC1)
        {
          //  if (k == 0)send(ClientSocket, (char*)&key, sizeof(key), 0);
```

```
    bool clientClose = false;
    char fileRequested[FILENAME_MAX];
    const int fileAvailable = 200;
    const int fileNotfound = 404;
    const int BUFFER_SIZE = 1024;
    char bufferFile[BUFFER_SIZE];

    std::ifstream file;
    do {
        int fileDownloaded = 0;
        memset(fileRequested, 0, FILENAME_MAX);
        /*std::cout << "Enter file name: " << std::endl;
        std::cin >> fileRequested, FILENAME_MAX;  */
        if (strcpy_s(fileRequested, FILENAME_MAX, "Datatem.txt") != 0) {
            cout << "Errrr";

        }

        byRecv = send(ClientSocket,(char*)&fileRequested, FILENAME_MAX, 0);
        if (byRecv == 0 || byRecv == -1) {
            clientClose = true;
            break;
        }

        file.open(fileRequested, std::ios::binary);
        if (file.is_open()) {
            int bySendinfo = send(ClientSocket, (char*)&fileAvailable, sizeof(int), 0);
            if (bySendinfo == 0 || bySendinfo == -1) {
                clientClose = true;
            }
            file.seekg(0, std::ios::end);
            long fileSize = file.tellg();
            bySendinfo = send(ClientSocket, (char*)&fileSize, sizeof(long), 0);
            if (bySendinfo == 0 || bySendinfo == -1) {
                clientClose = true;
            }
            file.seekg(0, std::ios::beg);
            do {
                file.read(bufferFile, BUFFER_SIZE);
                if (file.gcount() > 0)
                    bySendinfo = send(ClientSocket, (char*)&bufferFile, file.gcount(), 0);
                if (bySendinfo == 0 || bySendinfo == -1) {
                    clientClose = true;
                    break;
                }
            } while (file.gcount() > 0);

            file.close();
            std::cout << "File sended!!!\n";
            clientClose = true;
            key = 0;
            //if (k == 0)send(ClientSocket, (char*)&key, sizeof(key), 0);
        }
        else {
            int bySendCode = send(ClientSocket, (char*)&fileNotfound, sizeof(int), 0);
            if (bySendCode == 0 || bySendCode == -1) {
                clientClose = true;
                break;
            }
        }
    } while (!clientClose);

  }
//  mut.unlock();
```

```
        if (key==3) {
            recv(ClientSocket, (char*)&tt, sizeof(tt), 0);
            cout << "Set new control T ";
            sendT(tt);
            key = 0;
        }
        if (key == 4) {
            key = 0;
            menu = false;

        }
        if (key == 0xCC) {
            sendStop(check);
            cout << "system stoped\n";
            key = 0;
        }
        if (key == 0xEE) {
            sendStart(check);
            cout << "system started\n";
            key = 0;
        }

    }
    //tth3.detach();


}




// system("pause");
void COMTerminate(void);
CloseHandle(hCOMPort);


return 0;
}
```

ARdu.cpp

```
#include <windows.h>
#include <iostream>
#include <iostream>
#include <fstream>
#include <chrono>
#include <string>
#include "ARdu.h"

HANDLE hCOMPort = INVALID_HANDLE_VALUE;
OVERLAPPED ovlprd;
OVERLAPPED ovlpwr;
//=============================================================================
HANDLE hreader = INVALID_HANDLE_VALUE;
HANDLE hwriter = INVALID_HANDLE_VALUE;
DWORD WINAPI ReadThread(LPVOID);
DWORD WINAPI WriteThread(LPVOID);
void COMTerminate();
```

```
unsigned char bufrd[8], bufwr[1];
LPCTSTR sPortName = L"COM2";


using namespace std;
void sendStop(bool& i);
extern bool check;

//---------------------------------------------------------------------------
void COMTerminate(void) {
	if (hwriter != INVALID_HANDLE_VALUE) {
		TerminateThread(hwriter, 0);
		CloseHandle(ovlpwr.hEvent);
		CloseHandle(hwriter);
	}
	if (hreader != INVALID_HANDLE_VALUE) {
		TerminateThread(hreader, 0);
		CloseHandle(ovlprd.hEvent);
		CloseHandle(hreader);
	}
	hreader = INVALID_HANDLE_VALUE;
	hwriter = INVALID_HANDLE_VALUE;
}
//---------------------------------------------------------------------------
void COMPortStartThreads(void) {
	PurgeComm(hCOMPort, PURGE_RXCLEAR);
	hreader = CreateThread(NULL, 0, ReadThread, NULL, 0, NULL);
	hwriter = CreateThread(NULL, 0, WriteThread, NULL, CREATE_SUSPENDED, NULL);
}
//---------------------------------------------------------------------------
DWORD WINAPI WriteThread(LPVOID) {
	DWORD temp, signal;
	ovlpwr.hEvent = CreateEvent(NULL, true, true, NULL);
	while (1) {
		WriteFile(hCOMPort, bufwr, 1, &temp, &ovlpwr);
		signal = WaitForSingleObject(ovlpwr.hEvent, INFINITE);
		if ((signal == WAIT_OBJECT_0) && (GetOverlappedResult(hCOMPort, &ovlpwr, &temp, true))) {
		}
		else {
		}
		SuspendThread(hwriter);
	}
}
//---------------------------------------------------------------------------
DWORD WINAPI ReadThread(LPVOID) {
	DWORD btr, temp, mask, signal;
	ovlprd.hEvent = CreateEvent(NULL, true, true, NULL);
	SetCommMask(hCOMPort, EV_RXCHAR);
	while (1) {
		WaitCommEvent(hCOMPort, &mask, &ovlprd);
		signal = WaitForSingleObject(ovlprd.hEvent, INFINITE);
		if (signal == WAIT_OBJECT_0) {
			if (GetOverlappedResult(hCOMPort, &ovlprd, &temp, true)) {
				if ((mask & EV_RXCHAR) != 0) {
					ClearCommError(hCOMPort, &temp, &comstat);
					btr = comstat.cbInQue;

					if (btr) {

						ReadFile(hCOMPort, bufrd, btr, &temp, &ovlprd);
						ReadCOM(check);

						memset(bufrd, 0, 8);
					}
				}
```

```
                        }
                }
        }
}
//---------------------------------------------------------------------------
void ReadCOM(bool& i) {
        //SYSTEMTIME s;
        //GetLocalTime(&s);
        SYSTEMTIME s;
        GetLocalTime(&s);
        if ((i == true) && (bufrd[0] < 128)) {
                cout << "T=" << std::to_string(bufrd[0]) << "\n";
                ofstream out("Datatem.txt", std::ios::app);
                if (out.is_open())
                {
                        out << "T=" << std::to_string(bufrd[0]) << "\n";;
                        out << " " << s.wHour << ":" << s.wMinute << " " << s.wDay << "." << s.wMonth << "." <<
s.wYear << endl;
                        out.close();
                }


        }
}
//---------------------------------------------------------------------------
bool conCom() {
        DCB dcb;
        COMMTIMEOUTS timeouts;
        hCOMPort = CreateFile(sPortName, GENERIC_READ | GENERIC_WRITE, 0, NULL,
                OPEN_EXISTING, FILE_FLAG_OVERLAPPED, NULL);

        if (hCOMPort == INVALID_HANDLE_VALUE) {
                wcout << "error CreateFile(sPortName) " << sPortName;
                return false;
        }

        dcb.DCBlength = sizeof(DCB);

        if (!GetCommState(hCOMPort, &dcb)) {
                CloseHandle(hCOMPort);
                hCOMPort = INVALID_HANDLE_VALUE;
                wcout << "error GetCommState(hCOMPort) " << sPortName;
                return false;
        }
        dcb.BaudRate = CBR_9600;
        dcb.fBinary = TRUE;
        dcb.fOutxCtsFlow = FALSE;
        dcb.fOutxDsrFlow = FALSE;
        dcb.fDtrControl = DTR_CONTROL_DISABLE;
        dcb.fDsrSensitivity = FALSE;
        dcb.fNull = FALSE;
        dcb.fRtsControl = RTS_CONTROL_DISABLE;
        dcb.fAbortOnError = FALSE;
        dcb.ByteSize = 8;
        dcb.Parity = 0;
        dcb.StopBits = 0;
        if (!SetCommState(hCOMPort, &dcb)) {
                CloseHandle(hCOMPort);
                hCOMPort = INVALID_HANDLE_VALUE;
                wcout << "error SetCommState(hCOMPort) " << sPortName;
                return false;
        }

        timeouts.ReadIntervalTimeout = 10;
        timeouts.ReadTotalTimeoutMultiplier = 0;
```

```
		timeouts.ReadTotalTimeoutConstant = 0;
		timeouts.WriteTotalTimeoutMultiplier = 0;
		timeouts.WriteTotalTimeoutConstant = 0;

		if (!SetCommTimeouts(hCOMPort, &timeouts)) {
			CloseHandle(hCOMPort);
			hCOMPort = INVALID_HANDLE_VALUE;
			wcout << "error SetCommTimeouts(hCOMPort) " << sPortName;
			return false;
		}

		SetupComm(hCOMPort, 16, 16);

		PurgeComm(hCOMPort, PURGE_RXCLEAR);
		wcout << "connected COM port " << sPortName;
		return true;
}
//---------------------------------------------------------------------------
void sendStop(bool& i) {
		bufwr[0] = 0xCC;
		if (hwriter != INVALID_HANDLE_VALUE) {
			ResumeThread(hwriter);
		}
		i = false;
}
//---------------------------------------------------------------------------
void sendStart(bool& i) {
		bufwr[0] = 0xEE;
		if (hwriter != INVALID_HANDLE_VALUE) {
			ResumeThread(hwriter);
		}
		cout << "Started!" << endl;
		i = true;
}
//---------------------------------------------------------------------------
void sendT(int t) {
		bufwr[0] = (unsigned char)t;
		if (hwriter != INVALID_HANDLE_VALUE) {
			ResumeThread(hwriter);
		}
		cout << "Control T=" << std::to_string(t) << endl;
}
//---------------------------------------------------------------------------


//35    0x33 0x35 0x0A      0x23
```

ARdu.h

```
#pragma once

#ifndef Unit2H
#define Unit2H

void ReadCOM(bool& i);
void sendStop(bool& i);
void sendT(int t);
//void conComST(int t);
//int Check;
bool conCom();
void sendStart(bool& i);
```

```
void COMPortStartThreads(void);
void COMTerminate(void);

#endif
```