МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ УПРАВЛІННЯ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.
«_____»_____2021 р.

ДИПЛОМНИЙ ПРОЄКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ
«БАКАЛАВР»

Тема: «Система підтримки кліматичних показників у теплиці»

Виконавець: Гриугко Р.С.

Керівник: Масловський Б.Г.

Нормоконтролер: Тупота Є.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо-професійна програма «Системне програмування»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри

Литвиненко О.Є.
«_____»_____2021 р.

ЗАВДАННЯ
на виконання дипломної роботи (проекту)
Григурко Руслану Сергійовичу
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проекту) «Система підтримки кліматичних показників у теплиці»

затверджена наказом ректора від « 04 » лютого 2021 р. № 135/ст.

2. Термін виконання роботи (проекту): з 17 травня 2021 р. по 20 червня 2021 р.

3. Вихідні дані до роботи (проекту): програмна та технічна документація до *Iskra Neo*; редактор вихідного коду *Arduino* (*C++*); середовище конфігурації *Arduino IDE*; вимоги до оформлення дипломних проектів та ДСТУ 3008–95

4. Зміст пояснювальної записки:

Аналіз принципів побудови сучасних систем підтримки кліматичних показників.

Збірка виконуючого пристрою.

Програма системи керування виконуючим пристроєм.

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

1) структурна схема програми;

2) діаграма активностей;

3) схема алгоритму;

4) схема збірки системи.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1.	Провести огляд літератури за темою дипломного проекту та аналіз існуючих систем.	17.05.2021–18.05.2021	
2.	Зробити вибір середовища програмування і компонентів програмного комплексу	18.05.2021–20.05.2021	
3.	Розробити структуру програмних засобів системи	21.05.2021–22.05.2021	
4.	Розробити програмні засоби. Провести відладку програмних засобів	23.05.2021–05.06.2021	
5.	Написати пояснювальну записку	06.06.2021–10.06.2021	
6.	Підготувати графічний та ілюстративний матеріал	11.06.2021–12.06.2021	
7.	Підготувати доповідь та презентацію	12.06.2021–13.06.2021	

7. Дата видачі завдання: «_17_»___травня____ 2021 р.

Керівник дипломної роботи (проекту) _____ Масловський Б.Г.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Григурко Р.С.

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Система підтримки кліматичних показників у теплиці»: 42 сторінки, 11 рисунків, 17 літературних джерел, 3 таблиці та 1 додаток.

ARDUINO, МІКРОКОНТРОЛЕР *ATMEGA32U4*, ДАТЧИК *DHT-11*, КЛІМАТ-КОНТРОЛЬ, СЕРЕДОВИЩЕ КОНФІГУРАЦІЇ *ISKRA NEO*.

Об’єкт – сучасні засоби моніторингу та регулювання даних сенсорів.

Предмет – Система підтримки кліматичних показників у теплиці.

Мета дипломного проекту – програма зчитування даних з сенсорів та керування кліматичними показниками.

Методи проектування – мова програмування *Arduino*, редактор вихідного коду *Arduino IDE*, середовище конфігурації *Iskra Neo*.

Результати дипломного проектування рекомендується використовувати для підтримки кліматичних показників тепличних приміщень, а саме температури, вологості без участі людини, для забезпечення постійних оптимальних показників клімату.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

AVR – Alf and Vegard's RISC processor

SMD – Surface–Mount Device

IDE – Integrated Development Environment

USB – Universal Serial Bus

CDC – Communication Device Class

ВСТУП

Системи підтримки кліматичних показників (системи клімат-контролю) – система, що складається з кондиціонера, опалювальної системи, системи фільтрації, спеціальних датчиків, розташованих в різних місцях, а також електронного блоку управління кліматом. В найбільш просунутих системах може враховуватися не тільки температура, але навіть освітленість салону сонячними променями, що дозволяє забезпечити дійсно високий рівень комфорту. Якісна система клімат контролю враховує і вуличну температуру і рівень сонячного випромінювання.

Ринок подібних систем, включаючи обладнання, програмне забезпечення та послуги, динамічний, і основні сегменти зазнають швидких інновацій у галузі. Цей ринок зумовлений інноваціями у виробничих технологіях, необхідними переважно для автомобілебудування, «розумних будинках» та теплицях.

Системи клімат-контролю все частіше застосовуються у теплицях завдяки можливості постійної підтримки оптимального клімату. Більше того, завдяки різноманітному спектру датчиків, подібні системи легко налаштувати під свої потреби. Крім того, застосування систем клімат-контролю у теплицях має прямий вплив на якість та собівартість кінцевого продукту.

Потреби в постійній підтримці комфортних умов у середовищі, як у теплицях так і у автотранспорті та будинках, головним чином сприяє зростанню ринку подібних систем.

Системи клімат-контролю все більше застосовуються у повсякденному житті. Уже зараз такі системи є у більшості офісів та магазинах. Нові автомобілі створюються з вбудованою системою клімат-контролю.

Основним елементом системи управління клімат-контролем є електронний блок управління. Цей вузол приймає і обробляє інформацію, що надходить від датчиків, а також від панелі управління установкою, за допомогою якої користувач

встановлює температурний режим. На основі всіх отриманих даних ЕБУ здійснює управління виконавчими механізмами.

У більшості випадків клімат-контроль працює в автоматичному режимі – підтримує задану температуру, стежить за чистотою повітря. Але в багатьох версіях реалізовано також і ручне управління обігрівачем та кондиціонером, що дозволяє примусово подати потік повітря в потрібну зону.

Дані системи мають наступні переваги:

Автономність системи. Клімат-контроль підлаштовується під температурний режим і при необхідності скоректує потужність. Від користувача вимагається лише задати початкові параметри – температуру повітря і рівень вологості та освітлення.

Кілька режимів роботи. У моделей з автоматичним управлінням є режими, які підійдуть для різних цілей. Зазвичай в них передбачений режим що дозволяє швидко очистити повітря.

Налаштування для окремих приміщень. Якщо система охоплює декілька приміщень одразу можна задавати окремі параметри під кожне приміщення одночасно. Так, в одному може зберігатись прохолода, в той час як у сусідньому – тепло, що буде корисним для теплиць.

Налаштувати автоматичні програм. Е можливість налаштувати, щоб днем система працювала в економічному режимі, а ввечері включалася на повну потужність. Для внесення коригувань можна скористатися блоком управління або мобільним телефоном. Нові моделі підтримують управління з телефону.

Система клімат-контролю дозволяє користувачу створити комфортну атмосферу для далеких поїздок. На відміну від традиційного кондиціонера вона має зворотний зв'язок, що дозволяє в режимі реального часу коригувати вихідну температуру, ґрунтуючись на мінливих вхідних даних. А це, створює комфорт для людей, і мінімізує їх участь в управлінні системою.

Кліматична система надійна, економічна, зручна у використанні. Але у неї є недолік – висока ціна, в порівнянні з кондиціонером. Крім високої ціни на обладнання, покупець заплатить за монтаж – самостійно встановити систему не вийде.

На ринку доступний широкий асортимент мікроконтролерів різних виробників. Всі ці мікроконтролери мають унікальні функції та постачаються з різною ємністю оперативної пам’яті та ПЗП, різним набором інструкцій, різною архітектурою, регістрами тощо. Всі ці мікроконтролери відрізняються один від одного. Мікроконтролер є головним моментом кожного проекту, і успіх чи невдача проектів залежить від нього, і єдиний мікроконтролер не може бути використаний для кожної програми, оскільки кожна програма має різні вимоги. Тож вибір відповідного мікроконтролера – це завжди непросте завдання, оскільки слід врахувати ряд технічних особливостей.

Об’єкт проектування – підтримка кліматичних показників за допомогою зчитування з сенсорів стану температури та вологості і їх подальше коригування.

Предмет проектування – програмна система підтримки кліматичних показників.

Мета дипломного проекту – програмна система підтримки кліматичних показників.

Методи проектування – мова програмування *Arduino* (*C++*), редактор вихідного коду *Arduino IDE*, середовище конфігурації *Iskra Neo*.

Практичне значення отриманих результатів. Розроблена в даному дипломному проекті система підтримки кліматичних показників дозволяє створити та підтримувати оптимальні кліматичні умови без потреби у втручанні зі сторони людини.

РОЗДІЛ 1
ОГЛЯД ПРИНЦИПІВ ПОБУДОВИ СИСТЕМ ПІДТРИМКИ КЛІМАТИЧНИХ ПОКАЗНИКІВ

1.1. Основні поняття в системах підтримки кліматичних показників

Клімат-контроль – це система, що складається з кондиціонера, опалювальної системи, системи фільтрації, спеціальних датчиків, розташованих в різних місцях, а також електронного блоку управління кліматом. В найбільш просунутих системах може враховуватися не тільки температура, але навіть освітленість приміщень сонячними променями, що дозволяє забезпечити дійсно високий рівень комфорту. Якісна система клімат-контролю враховує і вуличну температуру і рівень сонячного випромінювання. [1]

Частіше за все подібні системи використовуються у теплицях, транспорті та «розумних будинках».

Система клімат-контролю у автомобілях зазвичай має наступні особливості:

− для запобігання запотівання скла одночасна праця обігрівача і кондиціонера;

− перед тим як потрапити в салон, повітря проходить фільтрацію, при високій забрудненості повітря, система автоматично переходить на рециркуляцію. [2]

Система клімат-контролю у «розумних будинках» має наступні особливості:

− окрім регулювання температури такі системи також регулюють рівень освітленості;

− також, системи «розумних будинків» мають систему безпеки. [3]

1.2. Принципи побудови систем підтримки кліматичних показників

За допомогою датчиків клімат-контроль відстежує температуру повітря зовні і всередині приміщення.

Потік ззовні нагнітається в вентилятором, розділяючись на 2 частини: одна проходить крізь випарник кондиціонера, а друга – через грубку. Після теплообмінників змішане повітря надходить через мережу дефлекторів і коробчатих повітропроводів.

Регулятор на панелі цього пристрою дозволяє встановити бажану температуру. Ці дані надійдуть в процесор, який буде керувати роботою системи. Так як внутрішні і зовнішні параметри постійно змінюються, їх відстеження відбувається в режимі реального часу.

Для виконання покладених на пристрій функцій в його конструкції є ряд компонентів:

- кондиціонер;
- панель управління;
- процесор;
- датчик температури положення заслінок;
- обігрівач;
- нагнітає вентилятор;
- система дефлекторів і повітропроводів;
- блок напрямних заслінок.

У кожного елемента системи своя функція.

Вентилятор створює тиск. В режимі рециркуляції повітря забирається системою як зсередини, так і зовні.

Блок заслінок монтується в одному корпусі з випарником кондиціонера і радіатором пічки. Така конструкція дозволяє привести параметри повітряного простору відповідно до встановлених значеннями.

Система дефлекторів і повітропроводів розподіляє потоки відповідно до певної програмою. Управління здійснюється за допомогою процесора, який отримує дані з панелі від перемикачів, регуляторів і датчиків.

Робота клімат-контролю безпосередньо залежить від стану його програми врядування та компонентів.

Це складна техніка, яка включає кондиціонер, датчики температури і вологості, обігрівач, електронний блок управління і систему фільтрації. Основна мета кліматичної системи – формування мікроклімату, а також його підтримку. Відповідними для людини визнані наступні умови: температура – від +22 до +25°*C*, вологість – від 65 до 80%. Клімат-контроль аналізує ці показники, а потім нагріває повітря (за допомогою грубки) або, навпаки, знижує його температуру (завдяки кондиціонеру). [4]

1.3. Аналіз ринку мікрококонтролерів для побудови системи

Оскільки системи клімат-контролю є пропрієтарними і постачаються тільки у комплекті з автомобілями або розумними будинками, а система розробляється під мікроконтролер проводиться аналіз ринку.

На ринку доступний широкий асортимент мікроконтролерів різних виробників. Всі ці мікроконтролери мають унікальні функції та постачаються з різною ємністю оперативної пам’яті та ПЗП, різним набором інструкцій, різною архітектурою, регістрами тощо. Всі ці мікроконтролери відрізняються один від одного. Мікроконтролер є головним моментом кожного проекту, і успіх чи невдача проектів залежить від нього, і єдиний мікроконтролер не може бути використаний для кожної програми, оскільки кожна програма має різні вимоги. Тож вибір відповідного мікроконтролера – це завжди непросте завдання, оскільки слід врахувати ряд технічних особливостей. Розглянуто деяких з основних постачальників.

PIC – сімейство мікроконтролерів, виготовлених *Microchip Technology*, похідне від *PIC*1650, спочатку розроблене відділом мікроелектроніки *General*

Instrument. Назва *PIC* спочатку називалася контролером периферійного інтерфейсу, і в даний час розширена як програмований інтелектуальний комп'ютер. Перші частини сім'ї були доступні в 1976 році; до 2013 року компанія відвантажила більше дванадцяти мільярдів окремих деталей, що використовуються в різноманітних вбудованих системах.

Ранні моделі *PIC* мали пам'ять лише для читання (ПЗУ) або програмований на місцях *EPROM* для зберігання програм, деякі з них мали можливість стирання пам'яті. Усі поточні моделі використовують флеш-пам’ять для зберігання програм, а новіші моделі дозволяють ПІК перепрограмувати себе. Пам'ять програм та пам'ять даних розділені. Пам'ять даних 8-розрядна, 16-розрядна і, в останніх моделях, 32-розрядна. Інструкції програми відрізняються за кількістю бітів залежно від родини *PIC* і можуть складати 12, 14, 16 або 24 біти. Набір команд також варіюється залежно від моделі, а більш потужні мікросхеми додають інструкції щодо функцій цифрової обробки сигналів.

Апаратні можливості пристроїв *PIC* варіюються від 6-контактних *SMD*, 8-контактних *DIP*-мікросхем до 144-контактних *SMD*-мікросхем, з дискретними виводами вводу-виводу, модулями *ADC* і *DAC*, а також портами зв'язку, такими як *UART*, *I2C*, *CAN* та навіть *USB*. Для багатьох типів існують малопотужні та високошвидкісні варіації.

Виробник постачає комп'ютерне програмне забезпечення для розробки, відоме як *MPLAB X*, асемблери та компілятори *C*/*C++*, а також апаратне забезпечення програмістів/тестерів у серії *MPLAB* та *PICKit*. Доступні також сторонні інструменти та деякі інструменти з відкритим кодом. Деякі деталі мають можливість програмування в ланцюзі; доступні як недорогі програмісти, так і високопродуктивні програмісти.

Пристрої *PIC* популярні як серед промислових розробників, так і серед любителів завдяки своїй низькій вартості, широкій доступності, великій базі користувачів, великій колекції приміток до програм, доступності недорогих або безкоштовних інструментів розробки, послідовному програмуванню та перепрограмованій можливості флеш-пам'яті. [5]

AVR – це сімейство мікроконтролерів, розроблене з 1996 року фірмою *Atmel*, придбане компанією *Microchip Technology* у 2016 році. Це модифікована архітектура Гарвардської 8-бітової одночіпової мікроконтролера *RISC*. *AVR* була однією з перших сімейств мікроконтролерів, яка використовувала вбудовану флеш-пам'ять для зберігання програм, на відміну від одноразових програмованих ПЗУ, *EPROM* або *EEPROM*, що використовувались на той час іншими мікроконтролерами.

Мікроконтролери *AVR* знаходять багато застосувань як вбудовані системи. Вони особливо поширені в програмах для любителів та вбудованих навчальних програм, що популяризуються завдяки включенню їх до багатьох ліній *Arduino* з відкритими платами розробки обладнання. [6]

Arduino – це апаратно-програмна компанія з відкритим вихідним кодом, спільнота проектів та користувачів, яка розробляє та виготовляє одноплатні мікроконтролери та набори мікроконтролерів для побудови цифрових пристроїв.

У конструкціях плат *Arduino* використовуються різноманітні мікропроцесори та контролери. Плати оснащені наборами цифрових та аналогових виводів вводу/виводу, які можуть бути з'єднані з різними платами розширення ("екрани") або платами (для прототипування) та іншими схемами. Плати мають послідовний інтерфейс зв'язку, включаючи універсальну послідовну шину (*USB*) на деяких моделях, які також використовуються для завантаження програм. Мікроконтролери можна програмувати за допомогою мов програмування *C* і *C* ++, використовуючи стандартний *API*, який також відомий як "мова *Arduino*". На додаток до використання традиційних ланцюжків інструментів компілятора, проект *Arduino* забезпечує інтегроване середовище розробки (*IDE*) та інструмент командного рядка (*arduino-cli*), розроблений в *Go*.

Проект *Arduino* розпочався в 2005 році як інструмент для студентів Інституту дизайну взаємодії *Ivrea* в Івреї, Італія, маючи на меті забезпечити недорогий і простий спосіб для початківців та професіоналів створювати пристрої, які взаємодіють із навколишнім середовищем за допомогою датчиків та виконавчі

механізми. Поширені приклади таких пристроїв, призначених для любителів-початківців, включають простих роботів, термостати та детектори руху. [7]

1.4. Висновки до розділу

Після аналізу програмних продуктів наявних на ринку, їх популярності та актуальності, а також сучасних засобів для клімат-контролю, сформульовано наступні задачі для виконання в ході дипломного проектування.

- на основі вихідної ідеї сформульовано завдання майбутнього проекту;
- розроблено концепцію проекту;
- проведено аналіз потрібності майбутнього продукту;
- проведено попередню оцінку ризиків майбутнього проекту;
- на основі концепції підготовано попереднє технічне рішення;
- вибрано методологію розробки, підготовлено попередній план робіт;
- вибрано сімейство мікроконтролерів під яку створюється система;
- проведено попередню оцінку трудовитрат і необхідних ресурсів;
- проведено аналіз можливості реалізації продукту.

РОЗДІЛ 2
СИСТЕМА ПІДТРИМКИ КЛІМАТИЧНИХ ПОКАЗНИКІВ У ТЕПЛИЦІ

2.1. Функціональні вимоги

До розроблюваної системи підтримки кліматичних показників поставлені наступні функціональні вимоги:

- моніторинг вологості повітря та ґрунту;
- моніторинг температури у теплиці;
- моніторинг освітленості теплиці;
- регулювання вологості повітря та ґрунту;
- регулювання температури у теплиці.

2.2. Структура інструментальних засобів розробки

Arduino – апаратна обчислювальна платформа для аматорського конструювання, основними компонентами якої є плата мікроконтролера з елементами вводу/виводу та середовище розробки *Processing/Wiring* на мові програмування, що є спрощеною підмножиною *C/C++*. *Arduino* може використовуватися як для створення автономних інтерактивних об'єктів, так і підключатися до програмного забезпечення, яке виконується на комп'ютері (наприклад: *Processing, Max/MSP, Pure Data, SuperCollider*). Інформація про плату (рисунок друкованої плати, специфікації елементів, програмне забезпечення) знаходяться у відкритому доступі і можуть бути використані тими, хто воліє створювати плати власноруч.

Плата *Arduino* складається з мікроконтролера *Atmel AVR*, а також елементів обв'язки для програмування та інтеграції з іншими пристроями. На багатьох платах наявний лінійний стабілізатор напруги +5*В* або +3,3*В*. Тактування здійснюється на частоті 16 або 8 *МГц* кварцовим резонатором. У мікроконтролер записаний завантажувач (*bootloader*), тому зовнішній програматор не потрібен.

На концептуальному рівні усі плати програмуються через *RS*–232 (послідовне з'єднання), але реалізація даного способу різниться від версії до версії. Новіші плати програмуються через *USB*, що можливо завдяки мікросхемі конвертера *USB–to–Serial FTDI FT232R*. У версії платформи *Arduino Uno* як конвертер використовується контролер *Atmega8* у *SMD*–корпусі. Дане рішення дозволяє програмувати конвертер таким чином, щоб платформа відразу розпізнавалася як миша, джойстик чи інший пристрій за вибором розробника зі всіма необхідними додатковими сигналами керування. У деяких варіантах, таких як *Arduino Mini* або неофіційній *Boarduino*, для програмування потрібно підключити до контролера окрему плату *USB–to–Serial* або кабель.

Плати *Arduino* дозволяють використовувати значну кількість виводів мікроконтролера як вхідні/вихідні контакти у зовнішніх схемах. Наприклад, у платі *Decimila* доступно 14 цифрових входів/виходів, 6 із яких можуть генерувати ШІМ сигнал, і 6 аналогових входів. Ці сигнали доступні на платі через контактні площадки або штирові роз'єми. Також існує багато різних зовнішніх плат розширення, які називаються «*shields*» («щити»), які приєднуються до плати *Arduino* через штирові роз'єми.

Arduino IDE це багатоплатформовий додаток на *Java*, що включає в себе редактор коду, компілятор і модуль передачі прошивки в плату. Середовище розробки засноване на мові програмування *Processing* та спроектоване для програмування новачками, не знайомими близько з розробкою програмного забезпечення. Мова програмування аналогічна мові *Wiring*. Загалом, це *C++*, доповнений деякими бібліотеками. Програми обробляються за допомогою препроцесора, а потім компілюються за допомогою *AVR-GCC*.

Програми *Arduino* пишуться на мові програмування С або С++. Середовище розробки *Arduino* поставляється разом із бібліотекою програм «*Wiring*» (бере початок від проекту *Wiring*, який дозволяє робити багато стандартних операцій вводу/виводу набагато простіше).

Введений користувачем код вимагає лише двох основних функцій, для запуску ескізу та основного циклу програми, які компілюються та зв'язуються за допомогою програми–заглушки *main*() у виконувану циклічну виконавчу програму з ланцюжком інструментів *GNU*, що також входить до дистрибутиву *IDE*. *IDE Arduino* використовує програму *avrdude* для перетворення виконуваного коду в текстовий файл у шістнадцятковому кодуванні, який завантажується на плату *Arduino* програмою завантажувача в прошивці плати. За замовчуванням *avrdude* використовується як інструмент завантаження для перенесення коду користувача на офіційні дошки *Arduino*.

Arduino IDE є похідним від *Processing IDE*, однак, починаючи з версії 2.0, *Processing IDE* буде замінено на платформу *IDE Eclipse Theia* на основі коду *Visual Studio*.

З ростом популярності *Arduino* як програмної платформи інші постачальники почали впроваджувати власні компілятори та інструменти з відкритим кодом (ядра), які можуть створювати та завантажувати ескізи на інші мікроконтролери, які не підтримуються офіційною лінійкою мікроконтролерів *Arduino*. [8]

C++ – мова програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно−орієнтованої, узагальненої та процедурної. Розроблена Б'ярном Страуструпом в *AT&T Bell Laboratories* 1979 року та початково отримала назву «*C* з класами». Згодом Страуструп перейменував мову на *C++* у 1983 р. Базується на мові *C*. [9]

У 1990−х роках *C++* стала однією з найуживаніших мов програмування загального призначення. Мову використовують для системного програмування, розробки програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм таких як

відеоігри. *C++* суттєво вплинула на інші, популярні сьогодні, мови програмування: *C#* та *Java*.

В 1998 році мова *C++* була стандартизована Міжнародною організацією стандартизації під номером 14882:1998 – Мова Програмування *C++*. Поточний стандарт − *C++11*, він був прийнятий у 2011 році робочою групою *MOC* після десятирічної підготовки.

Стандарт *C++* на 1998 рік складається з двох основних частин: ядра мови і стандартної бібліотеки. Стандартна бібліотека *C++* увібрала в себе бібліотеку шаблонів *STL*, що розроблялася одночасно із стандартом. Зараз назва *STL* офіційно не вживається, проте в колах програмістів на *C++* ця назва використовується для позначення частини стандартної бібліотеки, що містить визначення шаблонів контейнерів, ітераторів, алгоритмів і функторів.

Стандарт *C++* містить нормативне посилання на стандарт *C* від 1990 року і не визначає самостійно ті функції стандартної бібліотеки, які запозичуються із стандартної бібліотеки *C*.

Поза тим, існує величезна кількість бібліотек *C++,* котрі не входять в стандарт. У програмах на *C++* можна використовувати багато бібліотек *C*.

Стандартизація визначила мову програмування *C++,* проте за цією назвою можуть ховатися також неповні, обмежені достандартні варіанти мови. Спочатку мова розвивалася поза формальними рамками, спонтанно, у міру завдань, що ставилися перед ним. Розвиток мови супроводив розвиток крос-компілятора *Cfront*. Нововведення в мові відбивалися в зміні номера версії крос-компілятора. Ці номери версій крос-компілятора розповсюджувалися і на саму мову, але стосовно теперішнього часу мову про версії мови *C++* не ведуть.

При створенні *C++* прагнули зберегти сумісність з мовою *C*. Більшість програм на *C* справно працюватимуть і з компілятором *C++*. *C++* має синтаксис, заснований на синтаксисі *C*.

Нововведеннями *C++* порівняно з *C* є:

- підтримка об'єктно-орієнтованого програмування через класи;
- підтримка узагальненого програмування через шаблони;

- доповнення до стандартної бібліотеки;
- додаткові типи даних;
- обробка винятків;
- простори імен;
- вбудовані функції;
- перевантаження операторів;
- перевантаження імен функцій;
- посилання і оператори управління вільно розподіленою пам'яттю.

Переваги мови *C++*.

Швидкодія. Швидкість роботи програм на *C++* практично не поступається програмам на *C*, хоча програмісти отримали в свої руки нові можливості і нові засоби.

Масштабованість. На мові *C++* розробляють програми для найрізноманітніших платформ і систем.

Можливість роботи на низькому рівні з пам'яттю, адресами, портами.

Можливість створення узагальнених алгоритмів для різних типів даних, їхня спеціалізація, і обчислення на етапі компіляції, з використанням шаблонів.

Підтримуються різні стилі та технології програмування, включаючи традиційне директивне програмування, ООП, узагальнене програмування, метапрограмування (шаблони, макроси).

Недоліки мови *C++*.

Наявність безліч можливостей, що порушують принципи типобезпеки приводить до того, що в *C++* програми може легко закрастися важковловима помилка. Замість контролю з боку компілятора розробники вимушені дотримуватися вельми нетривіальних правил кодування. По суті, ці правила обмежують *C++* рамками якоїсь безпечнішої підмови. Більшість проблем типобезпеки *C++* успадкована від *C*, але важливу роль в цьому питанні грає і відмова автора мови від ідеї використовувати автоматичне управління пам'яттю (наприклад, збірку сміття). Так візитною карткою *C++* стали вразливості типу «переповнювання буфера».

Погана підтримка модульності. Підключення інтерфейсу зовнішнього модуля через препроцесорну вставку заголовного файлу серйозно уповільнює компіляцію, при підключенні великої кількості модулів. Для усунення цього недоліку, багато компіляторів реалізують механізм прекомпіляції заголовних файлів.

Недостача інформації про типи даних під час компіляції.

Мова *C++* є складною для вивчення і для компіляції.

Деякі перетворення типів неінтуїтивні. Зокрема, операція над беззнаковим і знаковим числами видає беззнаковий результат.

Препроцесор *C++* дуже примітивний. Це приводить з одного боку до того, що з його допомогою не можна здійснювати деякі завдання метапрограмування, а з іншого, в наслідок своєї примітивності, він часто приводить до помилок і вимагає багато дій з обходу потенційних проблем. Деякі мови програмування мають набагато могутніші і безпечніші системи метапрограмування.

З кінця 1990-х в спільноті *C++* набуло поширення так зване метапрограмування на базі шаблонів. По суті, воно використовує особливості шаблонів *C++* в цілях реалізації на їхній базі інтерпретатора примітивної функціональної мови програмування, що виконується під час компіляції. Сама по собі ця можливість вельми приваблива, але, внаслідок вище згаданого, такий код вельми важко сприймати і зневажувати. Мови *Lisp/Scheme*, *Nemerle* і деякі інші мають могутніші і водночас простіші для сприйняття підсистеми метапрограмування. Крім того, в мові *D* реалізована порівнянна за потужністю, але значно простіша в застосуванні підсистема шаблонного метапрограмування.

Хоча декларується, що *C++* мультипарадигмова мова, реально в мові відсутня підтримка функціонального програмування. Частково, даний пропуск усувається різними бібліотеками що використовують засоби метапрограмування для розширення мови функціональними конструкціями, але якість подібних рішень значно поступається якості вбудованих у функціональні мови рішень. Такі можливості функціональних мов, як зіставлення зі зразком взагалі украй складно емулювати засобами метапрограмування. [10]

Клімат-контроль – це система, що складається з кондиціонера, опалювальної системи, системи фільтрації, спеціальних датчиків, розташованих в різних місцях, а також електронного блоку управління кліматом. В найбільш просунутих системах може враховуватися не тільки температура, але навіть освітленість салону сонячними променями, що дозволяє забезпечити дійсно високий рівень комфорту. Якісна система клімат контролю враховує і вуличну температуру і рівень сонячного випромінювання.

2.3. Структура програмного модуля

Структурна діаграма. Основне використанні структурної діаграми полягає у графічному представленні відносин між різними частинами програми. Завдяки цій діаграмі можна провести більш глибокий аналіз програми. Для графічного відображення модулів використовують прямокутники. Структурні діаграми представляють статичні аспекти системи. У ній підкреслюються речі, які повинні бути присутніми в системі, що моделюється. Оскільки структурні діаграми представляють структуру, вони широко використовуються для документування програмної архітектури програмних систем. Наприклад, схема компонентів описує розподіл програмної системи на компоненти та демонструє залежності між цими компонентами. Структурну діаграму зображено на рис. 2.1.

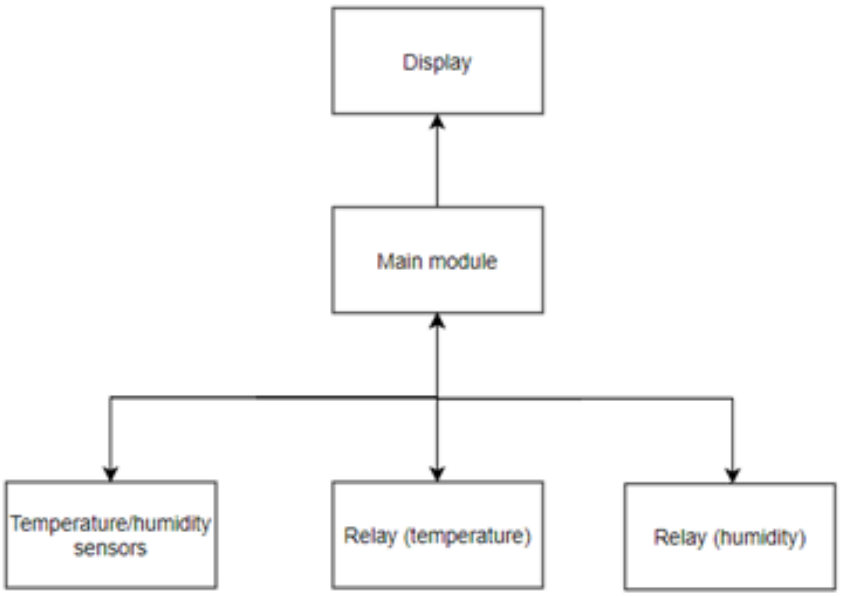

Рис. 2.1. Структурна схема комплексу

Діаграма діяльності – в *UML*, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Це графічні зображення робочих процесів поетапних дій та дій з підтримкою вибору, ітерації та паралельності. В уніфікованій мові моделювання діаграми діяльності призначені для моделювання як обчислювальних, так і організаційних процесів (тобто робочих процесів), а також потоків даних, що перетинаються з відповідними видами діяльності. Хоча діаграми діяльності в першу чергу показують загальний потік контролю, вони також можуть включати елементи, що відображають потік даних між діями через один або кілька сховищ даних. Діаграми діяльності можна розглядати як форму структурованої блок-схеми в поєднанні з традиційною діаграмою потоків даних. У типових техніках блок-схеми відсутні конструкції для вираження паралельності. Однак символи об’єднання та розділення на діаграмах діяльності вирішують це лише у простих випадках; значення моделі не зрозуміле, коли їх довільно поєднують з рішеннями або циклами. [11] Діаграма діяльності зображено на рис. 2.2.

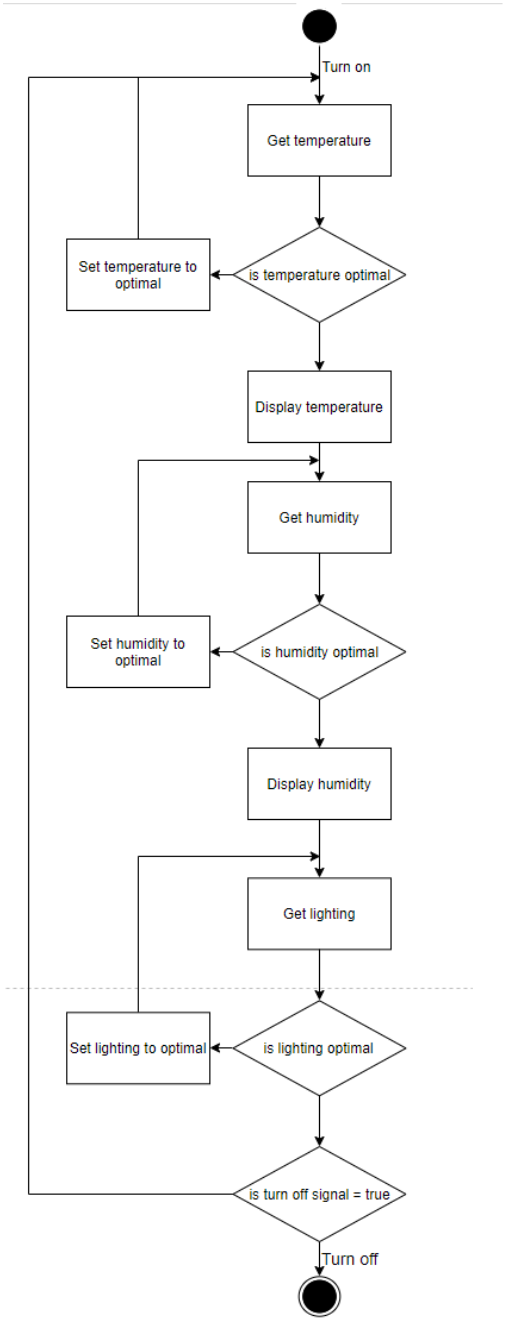

Рис. 2.2. Діаграма діяльностей

Схема алгоритму – це тип діаграми, що представляє робочий процес програми. Схема алгоритму також можна визначити як схематичне зображення алгоритму, поетапний підхід до вирішення завдання.

Схема алгоритму показує кроки як прямокутники різного типу та їх порядок, з'єднуючи прямокутники стрілками. Це схематичне зображення ілюструє модель рішення даної проблеми. Схема алгоритму використовуються для аналізу,

проектування, документування або управління процесом або програмою в різних областях. [12]

Алгоритм системи підтримки кліматичних показників виконує перевірку показників температури та вологості після чого подає сигнал на реле щодо зміни температури у разі невідповідності запрограмованих даних з даними що було отримано від сенсора. Схема алгоритму системи зображена рис. 2.3.

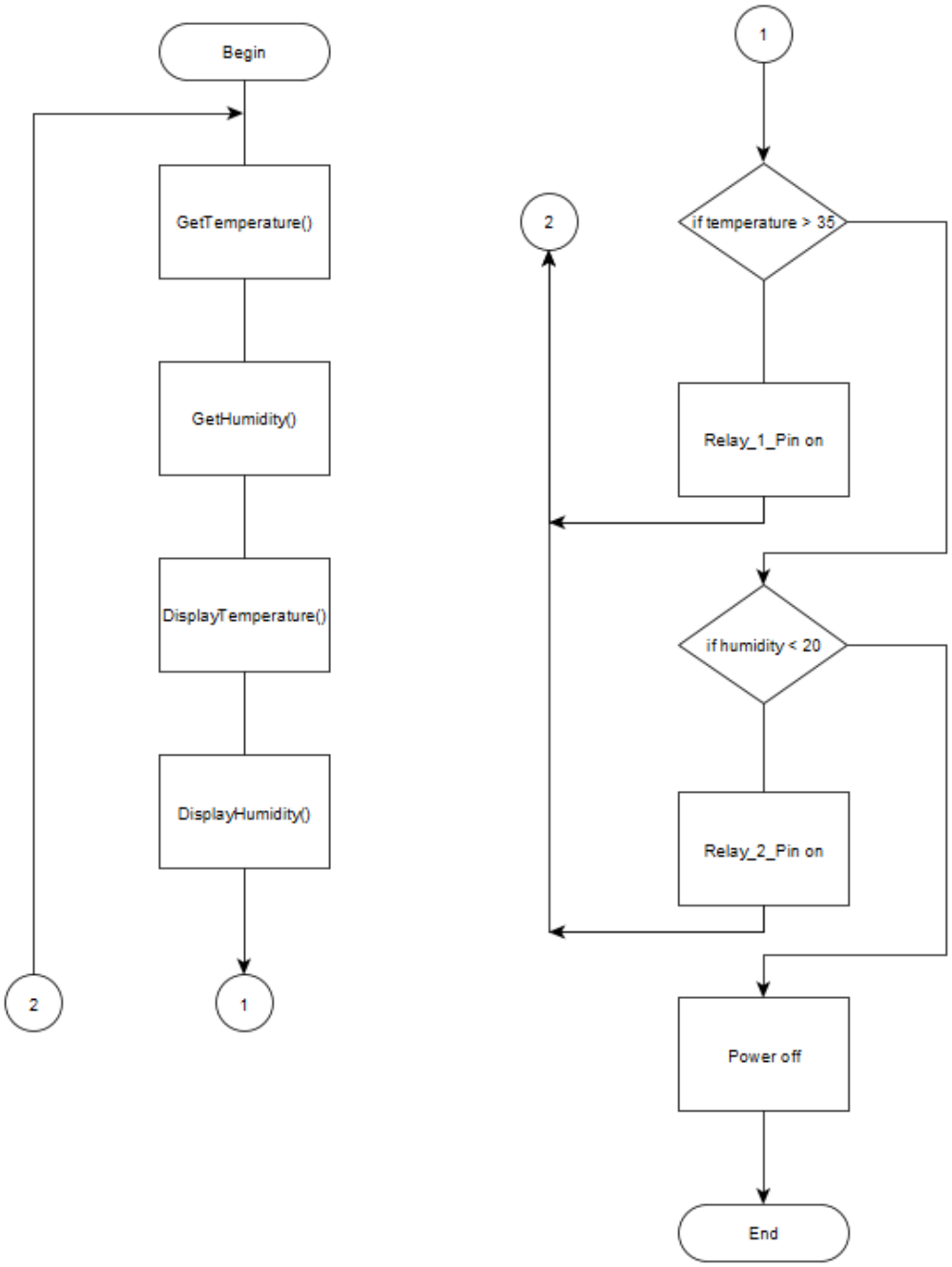

Рис. 2.3. Схема алгоритму системи підтримки кліматичних показників у теплиці

2.4. Збірка друкованої плати

В основі друкованої плати для системи підтримки кліматичних показників буде використовуватись мікроконтролер *ATmega32U4* в конфігурації з 32 кілобайтами флеш-пам’яті, 2.5 *Кбайт* статичної оперативної пам’яті з довільним доступом, 2 інтерфейси I^2C, та інтерфейс *UART* та 20 вводів\виводів загального призначення. На нього встановлюється *Troyka Slot shield v2*.

На платі розташовані шість *Troyka*-слотів з іменами від *A* до *F*. Кожен слот складається з чотирьох груп контактів, в які можна вставити один *Troyka*-модуль.

Сюди можна підключити:

- шість *Troyka*-модулів, керованих через один цифровий контакт;
- п'ять модулів, що використовують аналогові входи керуючої плати;
- до трьох модулів з інтерфейсом I^2C (*SDA / SCL*);
- до трьох модулів з інтерфейсом *SPI* (*DI / DO / CK*).

Точна кількість модулів, які можна підключити для роботи через інтерфейс *UART*, залежить від використовуваної вами плати управління.

Зчитування стану клімату буде виконуватись завдяки *DHT*-11.

DHT-11 – це датчик температури і вологості . Він призначений для роботи з температурами від 0 до 50 градусів по Цельсію з похибкою у 2 градуси. Діапазон вологості становить від 20% до 90% з похибкою у 5%. *DHT*-11 має невеликі габарити і працює при невеликих напругах. [13]

Вихідний калібрований цифровий сигнал *DHT*11. Він використовує ексклюзивну техніку збору цифрових сигналів і технологія зондування вологості, що забезпечує її надійність та стабільність. Його чутливими елементами є підключений до 8-бітного однокристального комп'ютера. Кожен датчик цієї моделі має температурну компенсацію та калібрування з точним калібруванням камери, а калібрувальний коефіцієнт зберігається в пам'яті *OTP*.

Невеликі розміри, низьке споживання та велика відстань передачі (20 *м*) дозволяють *DHT*11 бути пристосованим у всіх видах жорстких випадків

застосування. Однорядний упакований з чотирма шпильками, що робить підключення дуже зручне. Схему підключення драйверу зображено на рис. 2.4. [14]

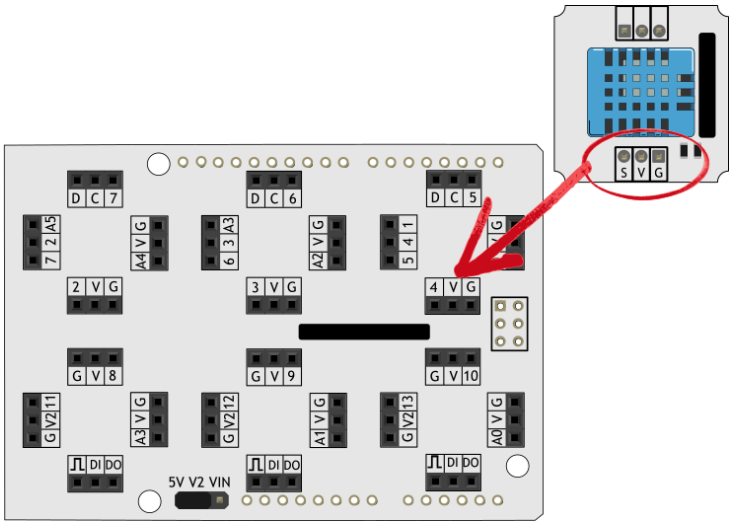

Рис. 2.4. Схема підключення *A4988*

Для підключення двох міні-реле на плату, їх потрібно розвернути на 90 градусів проти годинникової стрілки і вставити у вільні слоти верхнього ряду. Схему підключення реле зображено на рис. 2.5.

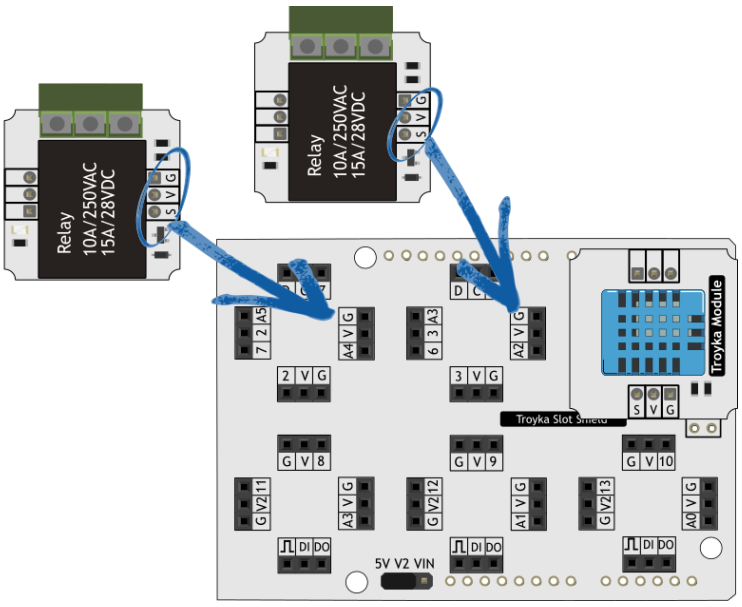

Рис. 2.5. Схема підключення реле

Чотирьохзарядний семисегментний індикатор з лінійки *Troyka*-модулів використовується в якості екрану для виводу показників. Цей модуль містить в собі чотири індикатори, при цьому для управління ними досить одного порту *SPI*. Модуль працює з керуючими сигналами напругою від 3 до 5*В*. Тому індикатор сумісний з більшістю плат, в тому числі з *Raspberry Pi*. Для встановлення модуля на платі, перевертаємо *Quad Display* на 180 градусів і підключаємо в лівий і центральний слоти нижнього ряду. Правильне підключення дисплею показано на рис.2.6.

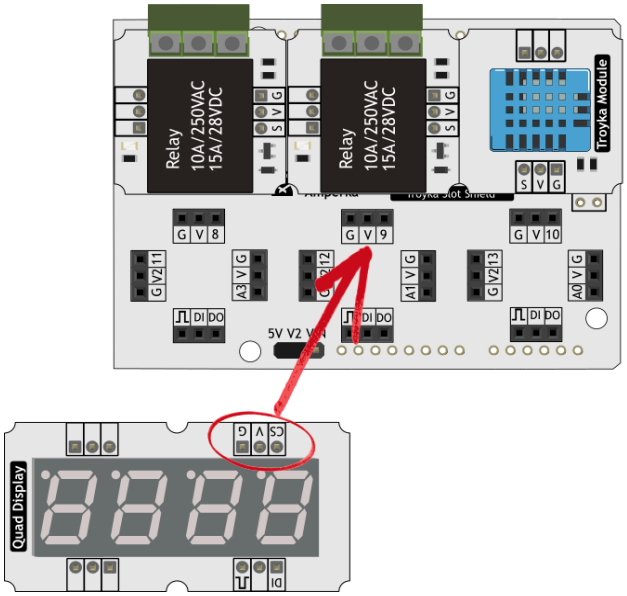

Рис. 2.6. Схема підключення дисплею

2.5. Висновки до розділу

Розглянуто основні функції сучасних систем клімат-контролю, що дозволило коректо сформулювати список функціональних вимог до модуля. Розроблено структуру програмного комплексу, що містить основні такі складові. Проаналізовано функціональні можливості різних типів середовищ розробки та інструментів для створення програмного модуля, обрано інструментальні засоби, що задовольняють потребам розробки системи моніторингу кліматичних показників.

РОЗДІЛ 3
ПРОГРАМА ПІДТРИМКИ КЛІМАТИЧНИХ ПОКАЗНИКІВ

3.1. Кроки створення програми

Для розробки була використана *Arduino IDE*. Для початку підлкючаємо бібліотеки сенсору та дисплею. Вказуємо номер цифрового піна що було зарезервовано під реле. Лістинг наведено нижче:

```
#include <TroykaDHT.h>
#include <QuadDisplay2.h>
#define RELAY_1_PIN  A4
#define RELAY_2_PIN  A2
```

Створюємо об’єкт класу *DHT*11 і передаємо номер піна до якого підключений датчик.

```
DHT dht(4, DHT11); QuadDisplay qd(9);
```

Створюємо змінну для зберігання стану системи, а також змінні для показників температури та вологості навколишнього середовища.

```
bool state = true;
float temperature = 0;
float humidity = 0;
```

Відкриваємо порт для моніторингу дій у програмі, починаємо роботу з датчиком *DHT*-11 та призначуємо два піни у режимі виходу.

```
long currentMillis = 0;
void setup(){
   Serial.begin(9600);
  qd.begin();
  dht.begin();
    pinMode(RELAY_1_PIN, OUTPUT);
  pinMode(RELAY_2_PIN, OUTPUT);
  currentMillis = millis();}
```

Прописуємо зчитування показників датчика кожні 3 секунди. Також, зчитуємо показники з датчика і виводимо їх на екран. Переключати що саме виводиться на екран – температура чи вологість можна натиснув на кнопку поруч з дисплеєм.

```
void loop(){
if (millis() – currentMillis > 3000) {
   state = !state;
   currentMillis = millis();
   dht.read();}
switch (dht.getState()) {
   case DHT_OK:
     temperature = dht.getTemperatureC();
     humidity = dht.getHumidity();
     if (state) {qd.displayTemperatureC(temperature);
     } else {
       qd.displayHumidity(humidity);}
     break;
```

У випадку помилки на дисплей виведеться повідомлення «*Err*».

```
    default:
      qd.displayDigits(QD_NONE, QD_E, QD_r, QD_r);
      break;}
  if (temperature > 35) {
    digitalWrite(RELAY_1_PIN, HIGH);
  } else {digitalWrite(RELAY_1_PIN, LOW);}
  if (humidity < 20) {digitalWrite(RELAY_2_PIN, HIGH);
  } else {digitalWrite(RELAY_2_PIN, LOW);}
  delay(5000);}

DHT::DHT(uint8_t pin, uint8_t type) {
    _pin = pin;
    _type = type;}

void DHT::begin() {}
```

Зчитування показників температури і вологості з датчика.

```
int8_t DHT::read() {
    uint8_t data[5];
    uint8_t dataBit;
    uint8_t checkSum;
    for (int i = 0; i < 5; i++)
        data[i] = 0;
    pinMode(_pin, OUTPUT);
    digitalWrite(_pin, LOW);
    delay(18);
    pinMode(_pin, INPUT_PULLUP);
```

Обробка помилок датчика, повернення стану з помилкою.

```
if (pulseInLength(_pin, HIGH, 40) == 0) {
   _state = DHT_ERROR_NO_REPLY;
   return _state;}

if (pulseInLength(_pin, LOW, 80) == 0) {
   _state = DHT_ERROR_NO_REPLY;
   return _state;}

if (pulseInLength(_pin, HIGH, 80) == 0) {
   _state = DHT_ERROR_NO_REPLY;
   return _state;}
```

3.2. Кроки прошивки мікроконтролера

Одні і ті ж частини коду часто використовуються в програмах різних типів. Наприклад, код для роботи з датчиком. Щоб не писати цей код кожного разу заново, його виносять в окремі файли – бібліотеки. Величезна кількість готового коду вже написано іншими людьми, і за допомогою бібліотек його можна легко використовувати в своїх програмах.

Дуже багато бібліотек йде в складі *Arduino IDE*. Додати бібліотеку в свій код можна з меню Ескіз –> Імпорт бібліотек –> Назва бібліотеки. Це показано на рис. 3.1.

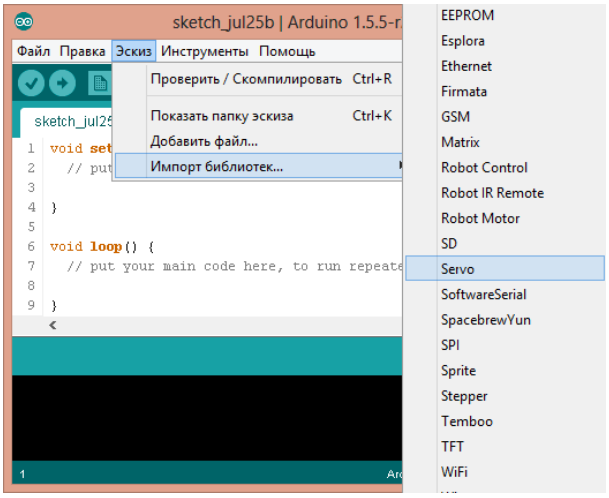

Рис. 3.1. Шлях до списку бібліотек

Бібліотек для *Arduino* дійсно дуже багато. І тільки незначна частина з них входить до складу *Arduino IDE*. Після завантаження архіву сторонньої бібліотеки потрібно зайти в *Arduino IDE* і виконати імпорт бібліотеки в вашу робочу папку *Sketchbook / libraries*, в якій повинні знаходиться все сторонні бібліотеки. Зробити це можна прямо з *Arduino IDE*, в меню Ескіз –> Імпорт бібліотек... –> Додати бібліотеку ... (рис 3.2.)

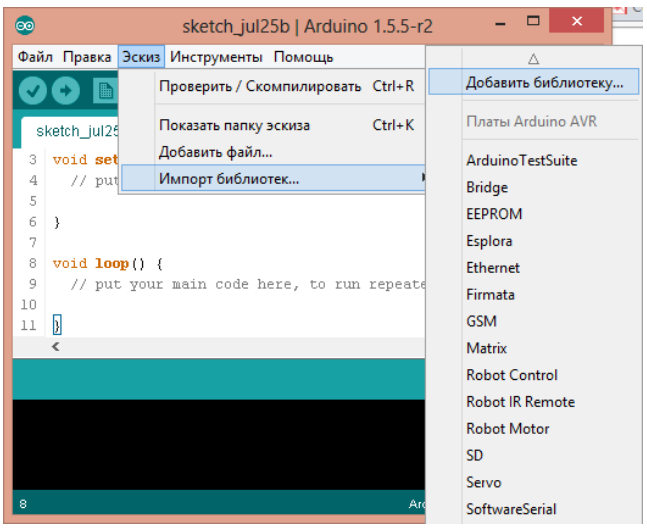

Рис. 3.2. Шлях до додавання бібліотеки

Відкриється діалогове вікно, в якому необхідно вибрати наш архів і натиснути кнопку *Open*.

Програмування *Iskra Neo*.

На відміну від «канонічного» *Arduino Uno, Iskra Neo* виконує підключення *USB* через виділений послідовний інтерфейс *CDC*. Це означає, що він не конфліктує з контактами 0 і 1 під час завантаження програми, що значить що потреба від'єднувати периферійне обладнання від контактів перед завантаженням відсутня.

З точки зору комп'ютера, *Iskra Neo* – це послідовний пристрій. Наприклад, він відображається як віртуальний *COM*-порт у Windows. Що цікавого в контролері *ATmega32U4*, це також може бути *HID* (*Human Interface Device*). Таким чином, ваш пристрій може легко прикинутися клавіатурою, мишею або геймпадом, якщо хочете. Далі наведено приклад програмування *Iskra Neo* з використанням *Arduino IDE*.

При використанні інсталятора, *Windows* – від *XP* до 10 – автоматично встановить драйвери, як тільки ви підключите плату. Після цього, клацніть на меню «Пуск» і відкрийте Панель управління. Перебуваючи на панелі керування, перейдіть до розділу Система та безпека. Далі клацніть на Система. Після того, як вікно системи відкриється, відкрийте диспетчер пристроїв. Подивіться на Порти (*COM & LPT*). Ви повинні побачити відкритий порт з назвою «*Iskra Neo* (*COMxx*)». Якщо немає розділу *COM & LPT*, у розділі «Інші пристрої» знайдіть «Невідомий пристрій». Клацніть правою кнопкою миші на порту «*Iskra Neo* (*COmxx*)» і виберіть опцію «Оновити програмне забезпечення драйвера». Далі виберіть опцію «Переглянути мій комп'ютер на наявність драйверів». Нарешті, перейдіть і виберіть файл драйвера з назвою «*arduino.inf*», який знаходиться в папці «Драйвери» програми для завантаження програмного забезпечення *Arduino* (а не в підкаталог «Драйвери *USB FTDI*»). Якщо ви використовуєте стару версію *IDE* (1.0.3 або старішу), виберіть файл драйвера *Uno* з назвою «*Arduino UNO.inf*» *Windows* закінчить установку драйвера звідти.

У *Arduino IDE*, відкрийте скетч який потрібно завантажити. Після цього, виберіть тип плати та порт. Потрібно буде вибрати запис в меню Інструменти> Плата, який відповідає вашій платі *Arduino*. Це показано на рис. 3.3.

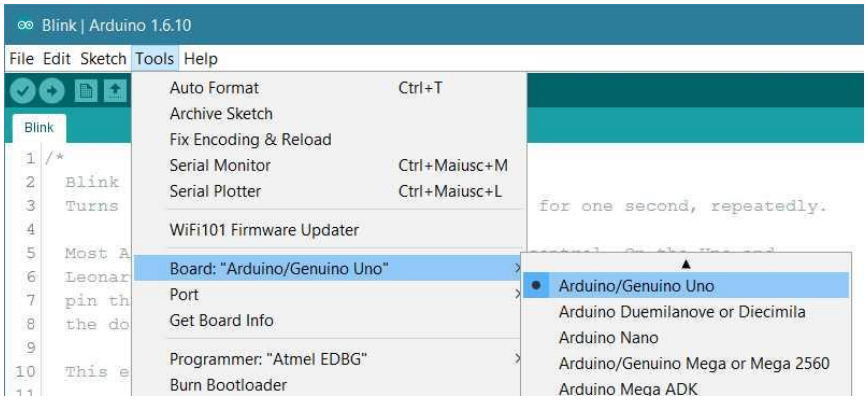

Рис. 3.3. Шлях вибору плати

Виберіть послідовний пристрій плати в Інструментах | Меню послідовного порту. Це, ймовірно, *COM3* або вище (*COM1* і *COM2* зазвичай зарезервовані для апаратних послідовних портів). Щоб це дізнатись, ви можете від'єднати свою плату та знову відкрити меню; запис, який зникає, повинен бути платою *Arduino*. Знову підключіть плату та виберіть послідовний порт, як показано на рис. 3.4.

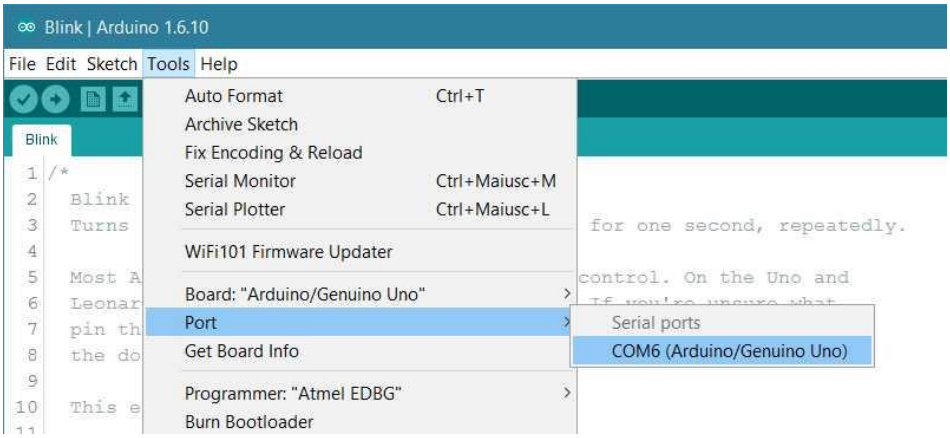

Рис.3.4. Шлях вибору портів

Після цього, натисніть кнопку «Завантажити» (рис. 3.5) в середовищі. Зачекайте кілька секунд – на платі повинні блимати світлодіоди *RX* і *TX*. Якщо завантаження завершиться успішно, з'явиться повідомлення «Завершення завантаження». з'явиться в рядку стану.

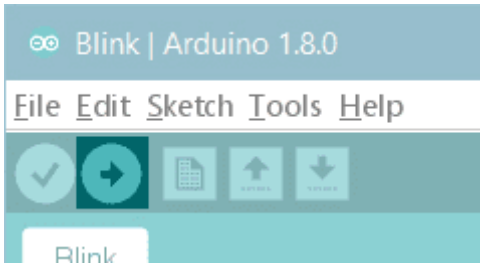

Рис.3.5. Кнопка завантаження

Через кілька секунд після завершення завантаження ви повинні побачити, як світлодіодний штифт 13 (*L*) на платі починає блимати (помаранчевим). Після чого мікроконтролер готовий до роботи.

3.3. Тестування

Вимоги до спроектованої системи, що будуть тестуватись:

- загальне включення та виключення системи;
- замірювання та відображення температури;
- замірювання та відображення вологості;
- зміна температури з часом на встановлену у коді;
- зміна вологості з часом на встановлену у коді;

Методи тестування.

За знанням системи – *white box*. Тестувальник має доступ до коду програм і може писати код, який пов’язаний з бібліотеками програмного забезпечення, що тестується.

За критерієм запуску програми – динамічне. Динамічні методи застосовуються в процесі безпосереднього виконання програми. Коректність програмного засобу перевіряється на безлічі тестів або наборів підготовлених вхідних даних. При прогоні кожного тесту збираються та аналізуються дані про відмови та збої в роботі програми.

Тестування системи буде відбуватися у домашніх умовах за відсутності теплиці.

Необхідні ресурси.

Для виконання поставлених навчальних завдань необхідно мати наступні знання та вміння:

– знання і вміння застосування на практиці стандарту *IEEE*−829;

– знання різних типів тестування в тому числі функціонального.

Тестові випадки:

Таблиця 3.1

Загальне включення та виключення системи

Дія	Очікуваний результат	Результат
Включити систему.	Система включена та доступна для використання.	Пройдено
Очікувати відображення температури.	На екрані з’явиться температура.	Пройдено
Очікувати відображення вологості.	На екрані з’явиться вологість.	Пройдено
Вимкнути систему.	Здійснюється вимкнення.	Пройдено

Відображення температури та вологості на екрані підтверджує успішне виконання тестування відображення інформації, що дає нам можливість переходу до наступного тесту – замірювання та відображення температури.

Таблиця 3.2

Замірювання та відображення температури

Дія	Очікуваний результат	Результат
Включити систему.	Система включена та доступна для використання.	Пройдено
Очікувати відображення температури.	На екрані з’явиться температура.	Пройдено
Очікувати зміни у показниках температури	На екрані з’явиться температура.	Пройдено
Виключити систему.	Здійснюється вимкнення.	Пройдено

Відображення зміни температури на екрані підтверджує успішне виконання тестування, що дає нам можливість переходу до наступного тесту – тестуванню зміни вологості. Цей тест перевіряв роботу одної з двох основних функцій системи клімат-контролю.

Таблиця 3.3

Замірювання та відображення вологості

Дія	Очікуваний результат	Результат
Включити систему.	Система включена та доступна для використання.	Пройдено
Очікувати відображення вологості.	На екрані з’явиться вологість.	Пройдено

Закінчення таблиці 3.3

Очікувати зміни у показниках вологості.	На екрані з’явиться вологість.	Пройдено
Виключити систему.	Здійснюється вимкнення.	Пройдено

Відображення зміни вологості на екрані підтверджує успішне виконання тестування, на чому тестування завершується.

3.4. Висновки до розділу

Спроектовано та реалізовано основний код програми та основні функції системи, а саме: реалізація функції відображення та зміну температури, відображення та зміну вологості та зміну рівня освітлення.

Протестована система моніторингу кліматичних змін за допомогою методів ручного тестування, а саме набору тест-кейсів, це показало рівень якості та дозволило перейти до наступного етапу тестування. Дані тест-кейси включали в себе: тестування зміни інформації на дисплеї, тестування зміни температури та тестування зміни вологості.

Проаналізовано результати виконаних тестувань та визначено рівень якості системи, це забезпечило завершення етапу розробки та перехід до етапу підтримки системи моніторингу кліматичних змін.

ВИСНОВКИ

Після аналізу програмних продуктів наявних на ринку, їх популярності та актуальності, а також сучасних засобів для клімат-контролю, сформульовано наступні задачі для виконання в ході дипломного проектування.

- на основі вихідної ідеї сформульовано завдання майбутнього проекту;
- розроблено концепцію проекту;
- проведено аналіз потрібності майбутнього продукту;
- проведено попередню оцінку ризиків майбутнього проекту;
- підготовано попереднє технічне рішення на основі оцінки ризиків;
- вибрано методологію розробки та попередній план робіт;
- проведено попередню оцінку трудовитрат і необхідних ресурсів;
- проведено аналіз можливості реалізації продукту.

Розглянуто основні функції сучасних систем клімат-контролю, що дозволило коректо сформулювати список функціональних вимог до модуля. Розроблено структуру програмного комплексу, що містить основні такі складові. Проаналізовано функціональні можливості різних типів середовищ розробки та інструментів для створення програмного модуля, обрано інструментальні засоби, що задовольняють потребам розробки системи моніторингу кліматичних показників.

Спроектовано та реалізовано основний код програми та основні функції системи, а саме: реалізація функції відображення та зміну температури, відображення та зміну вологості та зміну рівня освітлення.

Протестовано систему моніторингу кліматичних змін за допомогою методів ручного тестування, а саме набору тест-кейсів, це показало рівень якості та дозволило перейти до наступного етапу тестування.

Проаналізовано результати виконаних тестувань та визначено рівень якості системи, це забезпечило завершення етапу розробки та перехід до етапу підтримки системи моніторингу кліматичних змін.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Клімат-контроль. Що таке клімат-контроль? [електронний ресурс] – *www.infocar.ua* – Режим доступу: *https://www.infocar.ua/term_climatecontrol.html* (дата звернення 16.05.2021) – Назва з екрана.

2) Клімат-контроль у автомобілі. [електронний ресурс] – *www.etlib.ru* – Режим доступу: h*ttps://etlib.ru/blog/585-klimat-kontrol-avtomobilya---chto-eto-takoe-i-kak-im-polzovatsya* (дата звернення 16.05.2021) – Назва з екрана.

3) Системи клімат-контролю. [електронний ресурс] – *www.autoleek.ru* – Режим доступу: *http://autoleek.ru/jelektrooborudovanie/sistemy-komforta/sistema-klimat-kontrolya.html* (дата звернення 16.05.2021) – Назва з екрана.

4) Принцип роботи клімат-контролю. [електронний ресурс] – *www.principraboty.ru* – Режим доступу: *https://principraboty.ru/princip-raboty-klimat-kontrolya/* (дата звернення 16.05.2021) – Назва з екрана.

5) Microcontroller Programming: Microchip PIC; Sanchez and Canton; CRC Press; 824 pages; 2006;

6) AVR Programming: Learning to Write Software for Hardware; Elliot Williams; Maker Media; 474 pages; 2014;

7) Programming Arduino: Getting Started With Sketches; 2nd Ed; Simon Monk; McGraw-Hill Education; 192 pages; 2016;

8) *Arduino docs*. [електронний ресурс] – *www.arduino.cc* – Режим доступу: *https://docs.arduino.cc* (дата звернення 16.05.2021) – Назва з екрана.

9) Герберт Шилдт. Полный справочник по *C++ = C++: The Complete Reference*. − 4−е изд. − М.: Вильямс, 2011. − *С. 800*. − *ISBN 978*−5−8459−0489−8.

10) Бьёрн Страуструп. Язык программирования *C++ = The C++ Programming Language* / Пер. с англ. − 3−е изд. − СПб.; М.: Невский диалект − Бином, 1999. − 991 с. − 3000 экз. − *ISBN* 5−7940−0031−7 (Невский диалект), *ISBN* 5−7989−0127−0 (Бином), *ISBN* 0−201−88954−4 (англ.).

11) Chonoles, Michael Jesse; James A. Schardt (2003). UML 2 for Dummies. Wiley Publishing

12) ISO 5807 (1985). Information processing – Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts. International Organization for Standardization.

13) *Digital-output relative humidity & temperature sensor/module* [електронний ресурс] – *www.dfrobot.com* – Режим доступу: *http://image.dfrobot.com/image/data/KIT0003/DHT11%20datasheet.pdf* (дата звернення 16.05.2021) – Назва з екрана.

14) *DHT*-11 датчик температури та вологості [електронний ресурс] – *www.xcraft.com* – Режим доступу: *https://xcraft.com.ua/datchik-temperatury-i-vlazhnosti-dht11* (дата звернення 16.05.2021) – Назва з екрана.

15) *ISO 6983-1:2009. Automation systems and integration – Numerical control of machines – Program format and definitions of address words – Part 1:Data format for positioning, line motion and contouring control systems.* – Введ. 1983, Ред. 2009, 26 с.

16) ДСТУ 3008-95 Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.

17) Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.

ДОДАТОК А

Лістинг коду системи моніторингу кліматичних показників у таблиці

```
#include <TroykaDHT.h>
#include <QuadDisplay2.h>
#define RELAY_1_PIN  A4
#define RELAY_2_PIN  A2
DHT dht(4, DHT11);
QuadDisplay qd(9);
bool state = true;
float temperature = 0;
float humidity = 0;
long currentMillis = 0;
void setup(){
  Serial.begin(9600);
  qd.begin();
  dht.begin();
  pinMode(RELAY_1_PIN, OUTPUT);
  pinMode(RELAY_2_PIN, OUTPUT);
  currentMillis = millis();}
 void loop(){
  if (millis() - currentMillis > 3000) {
    state = !state;
    currentMillis = millis();
    dht.read();}

  switch (dht.getState()) {
    case DHT_OK:
```

```
      temperature = dht.getTemperatureC();
      humidity = dht.getHumidity();
      if (state) {
        qd.displayTemperatureC(temperature);
      } else {
        qd.displayHumidity(humidity);
      }
      break;
    default:
      qd.displayDigits(QD_NONE, QD_E, QD_r, QD_r);
      break;}
   if (temperature > 35) {
    digitalWrite(RELAY_1_PIN, HIGH);
   } else {
    digitalWrite(RELAY_1_PIN, LOW);
   }
   if (humidity < 20) {
    digitalWrite(RELAY_2_PIN, HIGH);
   } else {
    digitalWrite(RELAY_2_PIN, LOW);}
   delay(5000);}
#include "QuadDisplay2.h"

const static uint8_t numerals[] = { QD_0, QD_1, QD_2, QD_3, QD_4, QD_5,
QD_6, QD_7, QD_8, QD_9 };
QuadDisplay::QuadDisplay(uint8_t pinCS) {
  _pinCS = pinCS;
  _useSPI = true;}
QuadDisplay::QuadDisplay(uint8_t pinCS, boolean useSPI) {
  if (useSPI) {
```

```
        _pinCS = pinCS;
        _useSPI = true;
    } else {
        _pinCS = pinCS;
        _pinSCK = SCK;
        _pinDI = MOSI;
        _useSPI = false;}
}
QuadDisplay::QuadDisplay(uint8_t pinCS, uint8_t pinMOSI, uint8_t pinSCK) {
    _pinCS = pinCS;
    _pinDI = pinMOSI;
    _pinSCK = pinSCK;}
void QuadDisplay::begin() {
    pinMode(_pinCS, OUTPUT);
    digitalWrite(_pinCS, LOW);
    if (_useSPI) {
        SPI.begin();
    } else {
        pinMode(_pinSCK, OUTPUT);
        pinMode(_pinDI, OUTPUT);
        digitalWrite(_pinSCK, LOW);
        digitalWrite(_pinDI, LOW);}
}
void QuadDisplay::end() {
    digitalWrite(_pinDI, LOW);
    digitalWrite(_pinSCK, LOW);
    digitalWrite(_pinCS, LOW);
    pinMode(_pinSCK, INPUT);
    pinMode(_pinDI, INPUT);
    pinMode(_pinCS, INPUT);}
```

```
void QuadDisplay::beginWrite() {
   digitalWrite(_pinSCK, LOW);
   digitalWrite(_pinDI, LOW);
   digitalWrite(_pinCS, LOW);}
void QuadDisplay::writeData(uint8_t data, uint8_t n) {
   shiftOut(_pinDI, _pinSCK, LSBFIRST, data);}
void QuadDisplay::writeData(uint32_t data, uint8_t n) {
   for (uint8_t i = n; i > 0; i--) {
      digitalWrite(_pinDI, (data & 1));
      digitalWrite(_pinSCK, HIGH);
      digitalWrite(_pinSCK, LOW);
      data >>= 1;}
}
void QuadDisplay::endWrite() {
   digitalWrite(_pinDI, HIGH);
   digitalWrite(_pinSCK, HIGH);
   digitalWrite(_pinSCK, LOW);
   digitalWrite(_pinCS, HIGH);
   digitalWrite(_pinDI, LOW);
   digitalWrite(_pinSCK, LOW);}
uint8_t QuadDisplay::reverse(uint8_t x) {
   uint8_t mask = x & 0x01;
   x = x >> 1;
   x = ((x & 0x55) << 1) | ((x & 0xAA) >> 1);
   x = ((x & 0xCC) >> 2) | ((x & 0x33) << 2);
   x = (x >> 4) | (x << 4);
   x = x | mask;
   return x;}
void QuadDisplay::setDots(uint8_t array[]) {
   for (int i = 1; i < 4; i++) {
```

```
        if (array[i] & 0x01 == 0) {
            array[i - 1] &= 0xFE;
            array[i] |= 0x01;}
    }
}
void QuadDisplay::displayDigits(uint8_t digit1, uint8_t digit2, uint8_t digit3,
    uint8_t digit4) {
    if (_useSPI) {
        uint8_t digitsArray[] = { digit1, digit2, digit3, digit4 };
        setDots(digitsArray);
        digitalWrite(_pinCS, LOW);
        for (int i = 0; i < 4; i++) {
            SPI.transfer(reverse(digitsArray[i]));}
        digitalWrite(_pinCS, HIGH);
    } else {
        beginWrite();
        writeData(digit1);
        writeData(digit2);
        writeData(digit3);
        writeData(digit4);
        endWrite();}
}
void QuadDisplay::displaySegments(uint32_t digits) {
    beginWrite();
    writeData(digits);
    endWrite();}
void QuadDisplay::displayClear() {
    displayDigits(QD_NONE, QD_NONE, QD_NONE, QD_NONE);}
uint8_t QuadDisplay::getBit(uint8_t byte, uint8_t number) {
    return ((byte >> (8 - number)) & 1);}
```

```
void QuadDisplay::displayInt(int val, bool padZeros, uint8_t dots) {
  uint8_t digits[4] = { 0xff, 0xff, 0xff, 0xff };
  if (!padZeros && !val) {
    digits[3] = numerals[0];
  } else {
    bool negative = val < 0;
    val = abs(val);
    int8_t i;
    for (i = 4; i--;) {
      uint8_t digit = val % 10;
      digits[i] = (val || padZeros) ? numerals[digit] : 0xff;
      val /= 10;
      if (!val && !padZeros) {
        break;}
    }
    if (negative) {
      digits[i - 1] = QD_MINUS;}
    if (getBit(dots, 8)) {
      digits[3] &= QD_DOT;}
    if (getBit(dots, 7)) {
      digits[2] &= QD_DOT;}
    if (getBit(dots, 6)) {
      digits[1] &= QD_DOT;}
    if (getBit(dots, 5)) {
      digits[0] &= QD_DOT;}
  }
  displayDigits(digits[0], digits[1], digits[2], digits[3]);}
void QuadDisplay::displayFloat(float val, uint8_t precision, bool padZeros) {
  uint8_t dot = 0b0001;
  while (precision) {
```

```
        val *= 10;
        --precision;
        dot <<= 1;}
    displayInt((int)val, padZeros, dot);}
void QuadDisplay::displayTemperatureC(int val, bool padZeros) {
    uint8_t digits[4] = { 0xff, 0xff, QD_DEGREE, QD_C };
    if (!padZeros && !val) {
        digits[1] = numerals[0];
    } else {
        bool negative = val < 0;
        val = abs(val);
        int8_t i;
        for (i = 2; i--;) {
            uint8_t digit = val % 10;
            digits[i] = (val || padZeros) ? numerals[digit] : 0xff;
            val /= 10;
            if (!val && !padZeros) {
                break;}
        }
        if (negative) {
            digits[max(0, i - 1)] = QD_MINUS;}}
    displayDigits(digits[0], digits[1], digits[2], digits[3]);}
void QuadDisplay::displayHumidity(int val, bool padZeros) {

    uint8_t digits[4] = { 0xff, 0xff, QD_DEGREE, QD_UNDER_DEGREE };
    if (!padZeros && !val) {
        digits[1] = numerals[0];
    } else {
        bool negative = val < 0;
        val = abs(val);
```

```
    int8_t i;
    for (i = 2; i--;) {
      uint8_t digit = val % 10;
      digits[i] = (val || padZeros) ? numerals[digit] : 0xff;
      val /= 10;
      if (!val && !padZeros) {
        break;}
    }
    if (negative) {
      digits[max(0, i - 1)] = QD_MINUS;}
  }
  displayDigits(digits[0], digits[1], digits[2], digits[3]);}
void QuadDisplay::displayScore(int hour, int minute, bool blink) {
  uint8_t digits[4] = { 0xff, 0xff, 0xff, 0xff };
  if (!hour) {
    digits[0] = numerals[0];
    digits[1] = numerals[0];
  } else {
    if (hour < 10) {
      digits[0] = numerals[0];}
    int8_t i;
    for (i = 2; i--;) {
      uint8_t digit = hour % 10;
      digits[i] = hour ? numerals[digit] : 0xff;
      hour /= 10;
      if (!hour) {
        break;}
    }
  }
  if (!minute) {
```

```
    digits[2] = numerals[0];
    digits[3] = numerals[0];
  } else {
    if (minute < 10) {
      digits[2] = numerals[0];}
    int8_t i;
    for (i = 4; i--;) {
      uint8_t digit = minute % 10;
      digits[i] = minute ? numerals[digit] : 0xff;
      minute /= 10;
      if (!minute) {
        break;}
    }
  }
  if (blink) {
    if (millis() - _startMillis > 500) {
      _state = !_state;
      _startMillis = millis();}
    if (_state) {
      digits[1] &= QD_DOT;
    } else {
      digits[1] |= ~QD_DOT;}
  } else {
    digits[1] &= QD_DOT;}
  displayDigits(digits[0], digits[1], digits[2], digits[3]);}
```