НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет електроніки
(повна назва інституту/факультету)

Кафедра мікроелектроніки
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри
_____ О.В. Борисов
(підпис) (ініціали, прізвище)

“___”_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності (спеціалізації) Мікро- та наноелектронні прилади і пристрої
(код та назва спеціальності)

на тему: Цифровий зволожувач повітря з режимом інгаляції

Виконав: студент 4 курсу, групи ДП-52
(шифр групи)

Юрчик Юрій Іванович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник Кутова Оксана Юріївна _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент доцент, к.т.н. Карплюк Євгеній Сергійович _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Реферат

В роботі описаний процес розробки цифрового зволожувача повітря з режимом інгаляції і наведені результати виконаної роботи, а саме принципова схема приладу і програмний код. Процес розробки представлений кількома етапами. Наведені дані про роль вологості і актуальність створення приладу для її збільшення, також наведені дані щодо ефективності парових інгаляцій для лікування захворювань дихальних шляхів. На основі цих даних підібраний загальний перелік компонентів, які необхідні для реалізації приладу.

Для основних компонентів приладу розглянуті різні варіанти реалізації, описані загальні принципи роботи, переваги та недоліки. На основі описів компонентів наведені переваги і недоліки кожної реалізації компонентів в приладі, що розробляється. На основі цих даних вибрані такі реалізації, які найефективніше задовольняють поставленим вимогам.

Були розроблені прототип приладу та програмне забезпечення для його управління. Для взаємодії між підключеними компонентами написана програма для Arduino.

Abstract

The paper describes the process of developing a digital air humidifier with inhalation mode and presents the results of the work performed, namely the principle diagram of the device and the program code. The development process is presented in several stages. The given data on the role of humidity and the relevance of the device to increase it, as well as data on the effectiveness of steam inhalations for the treatment of respiratory diseases. On the basis of these data, a general list of components that are necessary for the implementation of the device is selected.

For the main components of the device are considered different options for implementation, described the general principles of work, advantages and disadvantages. Based on descriptions of components, the advantages and disadvantages of each component implementation in the developing device are given. Based on these data, the following implementations are selected that most effectively meet the requirements.

The prototype of the device and the software for its management were developed. For an interconnection between the connected components, the program for Arduino is written.

Зміст

1. Вступ

Кількість водяної пари, яка присутня в повітрі, може впливати не тільки на особистий комфорт, але й на різні виробничі процеси в промисловому застосуванні. Наприклад, в напівпровідниковій промисловості, рівень вологості повинен контролюватись для забезпечення належної обробки пластин. Контроль вологості також важливий для інкубаторів, респіраторного обладнання, стерилізаторів та біологічних продуктів. Крім того, наявність водяної пари може впливати на різні хімічні, біологічні та фізичні процеси. У галузях промисловості вимірювання вологості часто є важливим, оскільки це може вплинути на здоров'я та безпеку персоналу, а також на вартість продукту. Рівень вологості є важливим елементом погодних повідомлень, оскільки він вказує на можливість роси, туману або опадів. Більш висока відносна вологість знижує ефективність охолодження тіла. Це відбувається тому, що сповільнюється випаровування зі шкіри. Внаслідок цього в літній час однакова температура повітря відчувається вищою в місцях з високою вологістю, ніж з низькою.

1.1. Вологість у природі і житті людини

Кількість водяної пари, необхідної для досягнення насичення, зростає з підвищенням температури. По мірі зниження температури повітря вона досягає точки насичення без додавання або втрати водної маси. Кількість водяної пари, що міститься в повітрі, може істотно змінюватися.

Широко застосовуються два види запису рівня вологості: абсолютна та відносна. Абсолютна вологість описує вміст води в повітрі і виражається або в грамах на кубічний метр, або в грамах на кілограм [1]. Відносна вологість, виражена у відсотках, вказує на поточний стан абсолютної вологості відносно максимальної вологості при тій же температурі.

Абсолютна вологість - загальна маса водяної пари, яка присутня в одиниці об’єму повітря. В даному випадку температура не враховується.

Абсолютна вологість в атмосфері коливається від нуля до приблизно 30 грамів на кубічний метр, коли повітря насичується при 30 °C (86 °F) [2].

Абсолютна вологість змінюється, коли змінюється температура повітря або тиск при нефіксованому об’ємі. Це робить її непридатною для хімічних інженерних розрахунків, наприклад, при сушці, де температура може значно змінюватися. Тому абсолютну вологість визначають діленням кількості водяної пари, яка міститься в даному об’ємі повітря при даній температурі, на об’єм повітря. Цю величину вимірюють в грамах на кубічний метр або в грамах на кілограм. Такі одиниці вимірювання підходять для розрахунків і практичного застосування [3].

Відносна вологість повітряно-водної суміші визначається як відношення поточної маси водяної пари в суміші до максимальної маси водяної пари, яка може міститися при даній температури в одиниці об’єму.

Відносна вологість є важливим параметром, який використовується в прогнозах погоди та звітах, оскільки вона є показником ймовірності опадів, роси або туману. У спекотну літню погоду підвищення відносної вологості підвищує величину температури повітря, яка відчувається, для людини та тварин, перешкоджаючи випаровуванню поту з шкіри.

Вологість відіграє важливу роль у житті організмів. Для життя тварин, що використовують потовиділення для регулювання внутрішньої температури тіла, висока вологість погіршує ефективність теплообміну за рахунок зниження швидкості випаровування вологи з поверхні шкіри.

Люди чутливі до вологого повітря, тому що людський організм використовує випарне охолодження як основний механізм регулювання температури. У вологих умовах швидкість випаровування зі шкіри нижче, ніж в сухих умовах. Оскільки люди сприймають швидкість передачі тепла від тіла, а не саму температуру, ми відчуваємо повітря теплішим, коли відносна вологість висока, ніж тоді, коли вона низька.

Кондиціонер зменшує дискомфорт, зменшуючи не тільки температуру, але і вологість. Нагрівання холодного зовнішнього повітря може знижувати

рівні відносної вологості в приміщенні нижче 30%, [4] що призводить до таких дискомфортних явищ, як суха шкіра, потріскані губи, сухі очі і надмірна спрага.

1.2. Вологість в електроніці

Електронні пристрої часто розраховані на роботу тільки при певних умовах вологості (наприклад, від 5% до 95%). У верхній частині діапазону вологість може збільшити провідність ізоляторів, що призводить до несправності. Низька вологість може зробити матеріали крихкими. Такий рівень вологості сприяє накопиченню статичної електрики, що може призвести до спонтанного вимкнення комп'ютерів при виникненні розрядів. Крім помилкової нестабільності, електростатичні розряди можуть призвести до пробою діелектрика в твердотільних пристроях, що призводить до незворотного пошкодження. Центри обробки даних часто контролюють рівні відносної вологості з цих причин [5].

1.3. Актуальність інгаляцій для лікування захворювань дихальних шляхів

Для лікування гострих респіраторних захворювань популярні тепловологі інгаляції. Для цього застосовують спеціальні парові інгалятори, а також домашні підручні засоби. Але ефективність і лікувальні властивості подібних інгаляцій досить низькі. На сьогоднішній день про інгаляційну техніку відомо досить багато, досліджені розміри частинок, на які необхідно подрібнити препарат, щоб він досяг необхідного органу і здійснив лікувальну дію. Таким чином частинки з діаметром більше 10 мкм осідають в ротоглотці; від 5 до 10 мкм – в глотці, гортані і трахеї; від 1 до 5 мкм – в нижніх дихальних шляхах; від 0,5 до 1 мкм – досягають альвеол; менше 0,5 мкм – не осідають на органах і виходять при видиху повітря.

До переваг інгаляційної терапії відносяться: створення високої концентрації лікувального препарату в легенях, відсутність біологічної трансформації препарату (він не зв’язується з білками крові, не модифікується в печінці і т.д.), для досягнення лікувального ефекту необхідна менша доза, що, відповідно, знижує негативну дію деяких препаратів (антибіотики) на інші органи людини [6].

В лікуванні обструктивних захворювань широко використовуються інгаляції. Небулайзери використовуються при захворюваннях і важкому перебігу бронхіальної астми та інших хронічних обструктивних захворювань легень (професійні хвороби легень, бронхіт, бронхоектатична хвороба, хронічний обструктивний бронхіт і т.п.). Часто використання небулайзерів є єдиним способом доставити деякі лікувальні препарати (муколітики, антибіотики) до ураженого органу. Терапія дітей до одного року, а також ослаблених і важкохворих пацієнтів часто ускладнена без використання інгаляційної терапії [6].

1.4. Постановка задачі

В даній дипломній роботі необхідно було спроектувати пристрій, який буде суміщувати в собі автоматичний зволожувач повітря та небулайзер. Для виконання функції автоматичного зволоження пристрій має містити датчик вологості, причому його і випаровувач необхідно розміщувати так, щоб потік пари значно не впливав на поточні покази, інакше зволоження буде зупинятися кожні кілька хвилин, поки локальна вологість поряд з датчиком не вирівняється з кімнатною. Для визначення температури рідини, якою здійснюється інгаляція, необхідно використовувати датчик температури. За сигналом з датчика інгаляція буде вимикатись при низькій або високій температурі препарату. На панелі управління повинні бути розміщені кнопки та екран для індикації поточного режиму роботи пристрою, поточних параметрів вологості та налаштування величини відносної вологості, яку

необхідно досягнути. В режимі зволоження пристрій повинен працювати на максимальній потужності, щоб якнайшвидше збільшити вологість в приміщенні. Режим інгаляції повинен передбачати налаштування тривалості інгаляції та можливість налаштування інтенсивності випаровування води (лікувального препарату). На екрані в режимі інгаляції повинні відображатись поточні дані: потужність випаровування у відсотках, час з початку інгаляції та час до її закінчення, поточна температура рідини в контейнері в градусах Цельсію. Основним компонентом пристрою повинен бути контролер типу Arduino. Отже, в наступних розділах необхідно вибрати відповідний пристрій для випаровування; датчики для вимірювання температури води, вологості повітря; контролер Arduino, який задовольняє поставлену задачу і можливості якого будуть використані максимально; забезпечити регулювання потужності випаровування; вибрати екран для індикації; вибрати вентилятор, який виштовхуватиме з контейнера зволожене повітря; забезпечити живлення контролера, датчиків, індикатора, випаровувача і вентилятора.

1.5. Висновок до розділу 1

Для контролю вологості у приміщеннях її вимірюють у відносних одиницях. Наведена роль вологості в природі, для підтримання здоров’я людини, а також для предметів побуту людини, а саме для електроніки. Розглянуто актуальність використання інгаляцій для лікування дихальних шляхів людини. Також наведений перелік компонентів, які необхідні для реалізації необхідних функцій, особливості пристрою, який необхідно спроектувати і завдання на наступні розділи.

2. Варіанти реалізації ключових блоків приладу

2.1. Контролер Arduino

2.1.1. Особливості і реалізації Arduino

Момент появи перших мікроконтролерів відкрив початок нової ери розвитку мікропроцесорної техніки. Наявність в одному корпусі більшості системних пристроїв робить мікроконтролери подібними до сучасних комп'ютерів. Arduino та його аналоги являють собою набори, які складаються з готового блока електроніки і програмного забезпечення. Блок електроніки в даному випадку є друкованою платою з встановленими мікроконтролером і мінімумом електронних компонентів, які необхідні для його роботи. Фактично блок електроніки Arduino є аналогом материнської плати сучасного комп'ютера. Він має роз'єми для підключення зовнішніх пристроїв, часто роз'єм для зв'язку з комп'ютером, за допомогою якого і здійснюється програмування пам'яті мікроконтролера. Особливістю мікроконтролерів ATmega фірми Atmel є можливість їх програмування без використання спеціальних програматорів. Тому все, що необхідно для створення нового електронного пристрою, - плата Arduino, провід зв'язку і комп'ютер. Розробник може використати готові плати розширення або підключити безпосередньо до Arduino необхідні компоненти. Всі інші зусилля, в такому випадку, будуть спрямовані на розробку і налагодження програми керування на мові програмування високого рівня. Варіанти використання Arduino обмежені лише можливостями мікроконтролера і наявного варіанту реалізації даної платформи.

Сімейство плат Arduino налічує більше десятка реалізацій, які відрізняються технічними характеристиками і для кожного проекту можна вибрати максимально підходящий варіант, можливості якого будуть відповідати поставленій задачі і будуть використані максимально. Зокрема в сімействі апаратних платформ Arduino представлені такі моделі:

- Due — платформа на основі 32-бітного мікропроцесора Cortex-M3 ARM SAM3U4E;
- Leonardo — платформа на мікроконтролері ATmega32U4;
- Uno — платформа на мікроконтролері ATmega328, одна з найпопулярніших на даний момент;
- Diecimila — платформа на мікроконтролері ATmega168;
- Nano — платформа на мікроконтролері ATmega168 або ATmega328, малогабаритна, може використовуватись як макет;
- Mega2560 — платформа на мікроконтролері ATmega2560 з використанням мікросхеми ATMega8U2 для послідовного зв'язку за допомогою USB-порту;
- Mega ADK — версія платформи Mega 2560 з підтримкою інтерфейсу USB-host для підключення телефонів на базі Android та інших пристроїв з інтерфейсом USB.
- Mega — платформа на мікроконтролері ATmega1280;
- Arduino Bluetooth — платформа на мікроконтролері ATmega328, її особливістю є наявність на платі модуля Bluetooth BT-V06, який можна використати для бездротового зв'язку і програмування пам'яті мікроконтролера.
- LilyPad — платформа на мікроконтролері ATmega168V або ATmega328V, розроблена спеціально для перенесення і зашивання в тканину.
- Fio — платформа на мікроконтролері ATmega328, використовується для бездротових застосувань. Fio має на платі роз'єм для підключення модуля радіо XBee, роз'єм для батарейки LiPo і вбудовану схему підзарядки.
- Pro — платформа на мікроконтролері ATmega168 або ATmega328, може бути використана як складова частина великих проектів.

- Pro Mini — платформа на мікроконтролері ATmega168 або ATmega328, як і платформа Pro, може бути використана як складова частина великих проектів, вимоги до яких низька ціна, малі розміри і високий функціонал [7].

Оскільки в даному проекті не використовуються інтерфейси Bluetooth, Ethernet, Wi-FI, USB для взаємодії з іншими пристроями, то доцільно розглянути лише найпростіші та найменші моделі Arduino і з них вибрати найкращий варіант для поточної розробки.

2.1.2. Arduino LilyPad

Платформа Arduino LilyPad призначена для інтегрування в одяг, вона може бути зашита у тканину з вбудованими джерелом живлення, датчиками, індикаторами та іншими пристроями. Основа платформи - мікроконтролер ATmega168V або ATmega328V. Вони являють собою обмежені версії ATmega168 або ATmega328 відповідно з нижчими характеристиками і низьким енергоспоживанням, що дозволяє довгий час працювати від автономних джерел живлення. На даній платі відсутня схема для програмування мікроконтролера, тому запис програми в пам’ять необхідно здійснювати зовнішнім програматором [8].

Таблиця 2.1. Характеристики Arduino LilyPad

Мікроконтролер	ATmega168V або ATmega328V
Робоча напруга	2,7 – 5,5 В
Рекомендована вхідна напруга	2,7 – 5,5 В
Цифрові входи/виходи	14 (з них 6 можуть бути виводами для ШІМ)
Аналогові входи	8
Максимальний постійний струм входів/виходів	40 мА

Продовження таблиці 2.1.

Розмір флеш-пам’яті	16 кілобайт (ATmega168) або 32 кілобайти (ATmega328), з них 2 кілобайти використовує завантажувач
Розмір ОЗП	1 кілобайт (ATmega168) або 2 кілобайти (ATmega328)
Розмір EEPROM	512 байтів (ATmega168) або 1024 байти (ATmega328)
Тактова частота	16 МГц

2.1.3. Arduino Mini

Основа платформи Arduino Mini - мікроконтролер ATmega328. Плата може отримувати живлення від плати-конвертера, кабеля FTDI або від джерела живлення 5 В через контакт VCC або від джерела живлення 7 – 9 В через контакт +9V. Аналогічно як і в LilyPad, на даній платі відсутня схема для програмування мікроконтролера, тому запис програм в пам’ять необхідно здійснювати зовнішнім програматором [8].

Таблиця 2.2. Характеристики Arduino Mini

Мікроконтролер	ATmega328
Робоча напруга	5 В
Рекомендована вхідна напруга	7– 9 В
Цифрові входи/виходи	14 (з них 6 можуть бути виводами для ШІМ)
Аналогові входи	8
Максимальний постійний струм входів/виходів	40 мА
Розмір флеш-пам’яті	16 кілобайт, з них 2 кілобайти використовує завантажувач
Розмір ОЗП	1 кілобайт
Розмір EEPROM	512 байтів
Тактова частота	8 МГц або 16 МГц (залежить від моделі)

2.1.4. Arduino Nano

Платформа Arduino Nano має малі розміри і може використовуватись для простих проектів. Основою платформи є мікроконтролери ATmega328 (Arduino Nano 3.0) чи ATmega168 (Arduino Nano 2.x).

Arduino Nano можна живити від USB Mini-B, зовнішнього джерела живлення з напругою від 7 до 20 В (через контакт 30) або джерела живлення зі стабільною напругою 5В (через контакт 27). При підключенні платформи до кількох джерел живлення, вона живиться від джерела з найбільшою напругою.

Характеристики платформи Arduino Nano наведені в табл. 2.3 [8].

Таблиця 2.3. Характеристики Arduino Nano

Мікроконтролер	ATmega168 або ATmega328
Робоча напруга	5 В
Рекомендована вхідна напруга	7-12 В
Максимальна вхідна напруга	6-20 В
Цифрові входи/виходи	14 (з них 6 можуть бути виводами для ШІМ)
Аналогові входи	8
Максимальний постійний струм входів/виходів	40 мА
Максимальний постійний струм виводу 3,3 В	50 мА
Розмір флеш-пам’яті	16 кілобайт (ATmega168) або 32 кілобайти (ATmega328), з них 2 кілобайти використовує завантажувач
Розмір ОЗП	1 кілобайт (ATmega168) або 2 кілобайти (ATmega328)
Розмір EEPROM	512 байтів (ATmega168) або 1024 байти (ATmega328)
Тактова частота	16 МГц

2.2. Датчики для вимірювання температури

На сьогоднішній день існують таки види датчиків для вимірювання температури:

- Термоелектричні (термопари);
- Терморезистивні;
- П’єзоелектричні.
- Акустичні;
- Пірометричні.

Згідно поставленої задачі, вимірювати необхідно температуру води, отже датчик повинен працювати в діапазоні від 0 до 100^{o}C. Датчики, робота яких заснована на основі безконтактних методів вимірювання температури, недоцільно використовувати для цього діапазону, оскільки вони значно дорожчі, порівняно з контактними і їх використання доцільне лише у випадках неможливості використання контактних чи необхідності отримання даних про температуру у реальному часі з мінімальною затримкою, яка спричинена конструкцією контактних датчиків. В даному випадку відсутні вимоги щодо високої точності і низької затримки вимірювань.

2.2.1. Термоелектричні датчики

Термоелектричні термометри використовують явище виникнення термоелектрорушійної сили (термоЕРС) в точці контакту двох металів чи сплавів, яка залежить від температури місця з’єднання кінців двох різнорідних провідників, які являють собою чутливий елемент термометра – термопару. За законом зміни термоЕРС і вимірявши її величину вимірювальним приладом, можна розрахувати температуру точки спаю.

Термоелектричний термометр має в собі два з’єднані на одному кінці, ізольовані між собою в інших місцях термоелектроди, захисного чохла і головки з затискачами, за допомогою яких датчик підключається до

вимірювальної лінії. Він є первинним перетворювачем, тому повинен працювати в парі з вторинним перетворювачем, який перетворить напругу в значення температури. Вторинним перетворювачем можуть бути мілівольтметри, цифрові прилади.

Термопари дозволяють вимірювати опір в діапазоні від -200 до 2500 ^{o}C. Основна сфера їх використання – вимірювання високих температур, де використання інших датчиків ускладнене або неможливе [9].

2.2.2. Терморезистивні датчики

Терморезистивні датчики – це прилади, в основу роботи яких покладене явище зміни електричного опору речовини зі зміною температури. Їх використовують при вимірюванні температур в інтервалі від -260 до 750°C. Короткочасно деякі датчики можна використати для вимірювання температур до 1000 °C. Більшість металів мають позитивний температурний коефіцієнт електричного опору, тобто при нагріванні їх опір збільшується. Це пов’язано з підвищенням розсіювання електронів на неоднорідностях кристалічної гратки, що спричинено збільшенням теплових коливань іонів біля своїх положень рівноваги. В напівпровідниках при нагріванні число електронів провідності зростає з підвищенням температури. Це спричиняє зменшення електричного опору напівпровідників при нагріванні. Абсолютна величина температурного коефіцієнта електричного опору в напівпровідників вища, ніж у чистих металів. Вимірювання за допомогою термометрів опору проводяться при безпосередньому їх контакті з вимірюваним середовищем [9]. За графіками залежності опору термометра від температури можна визначити температуру середовища, з яким він контактує. Переваги металевих термометрів опору наступні:

- можливість виробництва приладів майже на будь-який інтервал температур;
- висока точність вимірювань;

- можливість підключення кількох датчиків до одного вимірювального приладу за допомогою системи перемикачів;
- можливість їх використання з цифровими приладами.

Головним недоліком цього виду термометрів є необхідність їх підключення до постійного джерела струму, що за порівняно низького опору датчика і високої напруги живлення може спотворювати результати вимірювань.

Чутливість напівпровідникових терморезисторів значно вища, ніж металевих, що є головною їх перевагою, але це також обмежує діапазон вимірюваних температур. Це спричинено тим, що опір напівровідникових терморезисторів змінюється в десятки і, навіть, тисячі разів в діапазоні вимірювання, що призводить до складнощів у проектувані схем вимірювальних приладів. Напівпровідникові термометри дозволяють вимірювати температуру в діапазоні від -80 до 150 °С [10].

2.2.3. Кварцові датчики

Кварцовий термометр – це термометр, принцип дії якого заснований на температурній залежності резонансної частоти п’єзоелемента. Частота п’єзоелектричного кристалу має високу стабільність. Кварцовий термометр не підходить для вимірювань, які потребують високої точності. Це спричинено його гістерезисом. Однак він має низький дрейф з часом, менше 0,005 К за місяць. Подібні термометри можуть використовуватись з високою точністю для відносних вимірювань, оскільки мають високу роздільну здатність за температурою. Крім того, наявність частотного вихідного параметра дозволяє його використовувати в автоматичних схемах, які самостійно регулюють тепловий режим об’єктів [10].

2.3. Зволожувачі повітря і технології зволоження

Поширені наступні види випарювачів води:

- випарювачі гарячої пари;
- випарювачі холодної пари;
- ультразвукові випарювачі;
- компресорні.

2.3.1. Випарювачі гарячої пари

Принцип дії даних випарювачів заснований на гарячому випарюванні води, тобто це звичайний процес кип’ятіння води в результаті якого вода перетворюється на пару і насичує повітря вологою. До того ж зволоження здійснюється стерилізованою парою. У зволожуючих приладах схема роботи дуже схожа на роботу електрочайника: до електродів підводиться напруга, вода нагрівається, починає кипіти і випаровується. При повному випарюванні води коло розмикається і прилад автоматично вимикається. Безпечність експлуатації закладена в самій конструкції зволожувача. З точки зору пожежної безпеки подібні парові зволожувачі не несуть вищої небезпеки, ніж звичайні побутові чайники.

Парові зволожувачі вирізняються простотою конструкції і, порівняно, невисокою ціною. Але їх низька ціна в значній мірі урівноважується високим енергоспоживанням. Продуктивність таких пристроїв досить висока – вони за короткий строк здатні збільшити вологість повітря до 80 – 100%. Працюють з мінімальним шумом внаслідок кипіння води і при цьому гріють приміщення гарячою парою. Але якщо температура повітря в будинку висока, надлишок тепла можна розцінювати як недолік [11].

2.3.2. Випарювачі холодної пари

2.3.2.1. Ультразвукові випарювачі

Ультразвукові випарювачі – найбільш ефективні серед існуючих на даний час. Вода з ємності подається на пластину, яка вібрує з високою (ультразвуковою) частотою, де вода розбивається на дрібні частинки. Потік повітря, що створюється вентилятором, подає її в приміщення, де вона переходить в пароподібний стан.

Таким чином ультразвуковий випарювач дозволяє генерувати туман в домашніх умовах. Пара, яку випускає пристрій, тільки на вигляд здається гарячою, на дотик вона холодна, а також абсолютно безпечна для здоров'я людини і для рослин.

На відміну від парових випарювачів, перевагою ультразвукових є точний контроль вологості, невисока температура вихідної пари (залежить від температури води) і низький рівень шуму. В ультразвукових випарювачах рекомендовано використовувати демінералізовану або дистильовану воду. Продуктивність ультразвукових випарювачів 7 – 12 літрів за добу при споживаній потужності менше 40 – 50Вт [12].

Ультразвукові випарювачі використовують енергію високочастотних коливань п'єзокристалу, що робить їх роботу майже безшумною і скорочує час інгаляції. Вони працюють безшумно, оскільки п'єзокристал коливається в діапазоні кількох сотень кілогерц, тобто вище частоти, яку може сприйняти людське вухо. Фракція, яка потрапляє в нижні дихальні шляхи перевищує 90% при середньому розмірі частинок від 2 до 3 мкм. Це дозволяє їх використовувати для інгаляцій.

Завдяки цьому аерозолі досягають дрібних бронхів і бронхіол в більш високій концентрації, підсилюючи лікувальний ефект. Залишковий обсяг препарату після інгаляції менше 0,5 мл, що дозволяє розпилювати препарат з мінімальними втратами [6].

2.3.2.2. Компресорні випарювачі

Компресорні випарювачі розбивають рідину, яку необхідно випарити на дрібні частинки за допомогою сильного потоку повітря. Такий принцип розпилення спричиняє високий рівень шуму при роботі і великий розмір готових пристроїв. Цей вид випарювачів неефективний для використання в кімнатних зволожувачах, оскільки внаслідок вище перелічених недоліків вони створюють значний дискомфорт за постійної роботи. Їх використання актуальне лише для розпилення препаратів, призначених для інгаляцій.

2.4. Датчики вологості

Датчики вологості повітря за принципом роботи поділяються на:

- ємнісні;
- резистивні;

2.4.1. Ємнісні гігрометри

У простому варіанті датчики вологості на основі зміни ємності є конденсаторами з повітряним діелектриком між обкладками. Його діелектрична проникність залежить від величини вологості, з цього слідує, що зміна вологості повітря веде до зміни ємності конденсатора.

Складнішим варіантом ємнісного гігрометра є датчик з діелектриком, діелектрична проникність якого значно залежить від поточної вологості. Досліджуваний матеріал поміщають між обкладками конденсатора, а сам конденсатор підключають до коливального контуру та генератора частоти. Для початку вимірюється власна частота коливання контуру, потім частота контуру з речовиною між обкладками. За виміряною частотою визначається ємність датчика після внесення зразка для досліджень.

До переваг поточного виду датчиків відносяться лінійна залежність ємності від вологості, низька інерційність, невисока вартість, значна точність вимірювань.

Ємнісні датчика мають певні недоліки. Вони зі значною похибкою вимірюють вологість менше 0.5%, також досліджуваний зразок не повинен містити матеріали з високою діелектричною проникністю, оскільки це значно спотворить результати вимірювань. Також важливо, щоб досліджуваний зразок залишався незмінної форми [10].

2.4.2. Резистивні гігрометри

Конструктивно резистивний датчик складається з двох електродів на підкладці. Зверху електроди покриті шаром матеріалу з низьким опором, який значно змінюється при зміні вологості. Найчастіше використовують оксид алюмінію внаслідок його дешевизни і поширеності. Цей матеріал здатний адсорбувати порівняно значну кількість вологи і це призводить до значної зміни опору датчика. Опір можна визначити за законом Ома, вимірявши прикладену напругу і струм через датчик. Важливими перевагами датчиків резистивного типу є їх низька вартість, простота конструкції, логарифмічна залежність опору від вологості, що спрощує перетворення опір-вологість, малі розміри, стабільність характеристик у часі [13].

2.5. Висновок до розділу 2

На основі описаних в першому розділі ключових блоків пристрою здійснений огляд можливих реалізацій цих блоків. Зокрема були розглянути різні види плат Arduino, їх особливості. Наведені і описані деякі види датчиків температури, вологості, розглянуті різні види перетворювачів води в пару.

3. Вибір технологій і компонентів для реалізації функцій пристрою

3.1. Широтно-імпульсна модуляція

3.1.1. Керування потужністю за допомогою ШІМ

ШІМ може використовуватися для контролю величини електроенергії, що подається на навантаження без втрат, які виникають внаслідок керування подачею енергії резистивними засобами. Недоліками даної методики є те, що потужність, яка споживається навантаженням, є не постійною, а переривчастою і енергія, що подається на навантаження, також не є постійною. Однак навантаження може бути індуктивним, і з достатньо високою частотою, тому, при необхідності, з використанням додаткових пасивних електронних фільтрів імпульси можуть бути згладжені і буде отримана аналогова форма сигналу. Потік потужності в навантаження може бути безперервним. Потік електроенергії від джерела живлення є непостійним і в більшості випадків вимагатиме накопичення енергії на стороні постачання.

Високочастотні системи керування потужністю на основі ШІМ легко реалізуються за допомогою напівпровідникових комутаторів. Як сказано вище, мінімальна потужність розсіюється за допомогою перемикача в стані увімкнення або вимкнення. Однак під час переходів між станами включення і виключення напруга і струм є ненульовими і, отже, потужність розсіюється в перемикачах. Швидко змінюючи стан між повністю включеним і повністю вимкненим (зазвичай менше 100 наносекунд), розсіювання потужності в перемикачах може бути досить низьким порівняно з потужністю, що подається на навантаження [14].

Сучасні напівпровідникові комутатори, такі як МОН-транзистори або біполярні транзистори з ізольованим затвором, добре підходять для високоефективних контролерів. Частотні перетворювачі, що використовуються для керування двигунами змінного струму, можуть мати ефективність вище 98%.

Контролери вентиляторів з керованою швидкістю обертання зазвичай використовують ШІМ, оскільки це набагато ефективніше в порівнянні зі змінним резистором. До того ж, останній практично не може працювати в електронному вигляді, він потребує невеликого двигуна.

Легкі димери для домашнього використання використовують певний тип ШІМ-контролю. Домашні світлові димери зазвичай містять електронні схеми, які припиняють потік енергії протягом певних ділянок кожного циклу напруги мережі змінного струму. Регулювання яскравості світла, що випромінюється джерелом світла, здійснюється керуванням напругою.

Ці досить прості типи димерів можуть бути ефективно використані з інертними (або з відносно повільною реакцією) джерелами світла, такими як лампи розжарювання. Керування потужністю може викликати лише незначні додаткові коливання випромінювання світла. Деякі інші типи джерел світла, такі як світлодіоди, вмикаються і вимикаються надзвичайно швидко і можуть мерехтіти, якщо вони живляться напругою низьких частот. Можливі ефекти мерехтіння від таких джерел світла швидкого реагування можуть бути зменшені за рахунок збільшення частоти ШІМ. Якщо світлові флуктуації є досить швидкими, візуальна система людини не може їх сприймати, тоді око сприймає середню інтенсивність без мерехтіння [14].

3.1.2. Реалізація ШІМ на Arduino

Усі контролери Arduino мають вбудовані апаратні модулі ШІМ, тому для використання ШІМ достатньо підключити навантаження до виходу, який підтримує дану технологію і налаштувати відповідним чином програму. Arduino Nano з ATmega328 підтримує ШІМ на цифрових виходах 3, 5, 6, 9, 10, 11 з роздільною здатністю 8 біт, тобто на виході можна отримати 256 значень напруги від нуля до напруги логічної одиниці, чого достатньо для більшості завдань. Відповідно при напрузі логічної одиниці 5В крок зміни середньої

напруги за період ШІМ становить $5В/256 \approx 0{,}0195$ В $\approx 19{,}5$ мВ. Для роботи з ШІМ в Arduino використовується функція analogWrite(). Її синтаксис:

```
analogWrite(pin, value);
```

де “pin” – номер виходу, на якому необхідно отримати ШІМ сигнал; “value” – період робочого циклу: значення від 0 до 255. Це значення визначає середнє значення напруги за період сигналу ШІМ [8].

3.2. Підбір плати Arduino для приладу

Для поточної розробки не використовуються безпровідні інтерфейси, немає потреби у зв’язки з телефонами та іншою USB периферією, відсутні вимоги автономності і висока швидкодія, тому підійдуть найпростіші реалізації Arduino. Для розробки виберемо платформу Arduino Nano, оскільки вона з-поміж найпростіших платформ Arduino виділяється наявністю схеми програмування пам’яті мікроконтролера на основній платі і наявністю можливості підключення до персонального комп’ютера за допомогою USB, що дозволяє легко проводити розробку і налагодження пристрою без зовнішніх програматорів. 14 цифрових входів/виходів даної платформи достатньо для зв’язку і керування усіма електронними компонентами і пристроями, які будуть підключені до плати. Також плату можна підключити до зовнішнього джерела живлення з напругою до 20 вольт, що дозволяє уникнути розробки додаткового перетворювача напруги з виходом 5 В для Arduino. Це актуально, коли в схемі присутні пристрої з напругою живлення вище 5 В, наприклад, вентилятори, лампи розжарювання, світлодіодні стрічки і т.п.

3.3. Підбір екрану для виводу інформації

Для показу інформації можна використовувати семисегментні індикатори та рідкокристалічні (далі РК) екрани. Розглянемо переваги і

недоліки кожного виду.

До переваг сегментних індикаторів можна віднести відносну простоту керування ними, можливість читання з них при яскравому освітленні, вони дозволяють відображати усі цифри та більшість букв алфавіту. До недоліків відносяться їх громіздкість, принципова неможливість відображення інформації без випромінювання світла, необхідність наявності зовнішнього декодера.

До переваг РК дисплеїв відносяться можливість відображення будь-яких символів, можливість роботи без підсвітки, компактність, простота взаємодії з ними, оскільки достатньо передати на дисплей лише код символу і координату, вбудований контролер самостійно визначить пікселі, які необхідно засвітити.

Для даного приладу виберемо символьний РК дисплей з двома рядами для відображення, в кожному з яких можна відобразити 16 символів. Він цілком задовольняє поставлену задачу. 32 символів цілком достатньо для відображення пунктів меню та індикації параметрів. Цей вибір доцільний, оскільки семисегментні індикатори не дозволяють відображати спецсимволи типу значків градуса, відсотка. З іншої сторони прилад не передбачає відображення складної графічної інформації (рисунки, графіки і т.п.), тому вибір РК матриці високого розширення недоцільний.

В поточному приладі використаємо символьний РК модуль RC1602B виробництва компанії Raystar. Він працює на контролері ST7066 [15] і його можна замінити дисплеєм з таким же або аналогічним контролером.

3.4. Підбір датчика температури під задачу

Для поточної задачі з розглянутих датчиків термоелектричні не підходять, оскільки їх вимірюваний діапазон значно більший, ніж той, який необхідний для задачі. Це призведе до неефективного використання, до того ж ціна термопарних датчиків порівняно висока. Кварцовий датчик

використовується для відносних вимірювань в парі з іншими датчиками, а також він має високу роздільну здатність по температурі, значно вище, ніж необхідно для поставленої задачі, що веде до неефективного використання його можливостей і, відповідно, грошових ресурсів. Резистивні датчики цілком задовольняють вимоги до вимірювань. Оскільки вони бувають металеві і напівпровідникові, то необхідно вибрати вид.

Очевидно, що напівпровідникові кращі, ніж металеві в потрібному діапазоні, оскільки вони мають менший температурний діапазон, ніж металеві, і цілком задовольняють вимоги. Але більшість напівпровідникових датчиків випускаються у вигляді мікросхем без спеціального захисту від впливу агресивного середовища на їх контакти, тому варто надати перевагу металевим резистивним датчикам температури, оскільки вони завжди упаковуються в металевий корпус з металу, наприклад, в капсулу з нержавіючої сталі.

З представлених на ринку металевих резистивних датчиків температури вибраний датчик PT100. Він виготовлений з платини і має опір 100 Ом при температурі 0 °C. Вимірюваний діапазон температур від -20 °C до +500 °C з допуском відхилень до 0,3 °C у вищу сторону і 0,8 °C в нижчу. Цей терморезистор з позитивним коефіцієнтом опору, який рівний 0,39 Ом/°C, тобто збільшення опору датчика на кожні 0,39 Ом вище поточного опору свідчитимуть про відповідне зростання температури. Датчик має форму циліндра з діаметром 4 мм і висотою 30 мм, оболонка виготовлена зі сталі [16].

3.5. Вибір пристрою для випарювання

Вибраний пристрій випарювання повинен задовольняти необхідність ефективного зволоження і при цьому мати можливість для використання його в якості інгалятора. Випарювач гарячої пари ефективно зволожує, хоча і в цій ролі має недолік – побічним ефектом є нагрівання повітря і небезпека отримати опіки при недотриманні умов експлуатації. Як інгалятор він не може

бути використаний, оскільки пара на виході має високу температуру, яка недопустима при інгаляціях. Тому вибір залишається серед випарювачів холодної пари. Компресорний випарювач може бути використаний лише в якості випарювального пристрою інгалятора, оскільки має достатню для цього продуктивність і дозволяє використовувати будь-які препарати. З іншого боку він абсолютно не підходить для зволоження повітря, оскільки працює з високим рівнем шуму, який шкідливий за тривалого впливу на людину. Найповніше задовольняє всі вимоги ультразвуковий випарювач. Він має низький рівень шуму, який спричинений лише взаємодією дрібних частинок води. Його продуктивність достатня, щоб обслуговувати невеликі приміщення і при цьому він може працювати як інгалятор. Останнє використання випливає з того, що розмір розпилених частинок становить 2-3 мкм, що дозволяє цим частинкам пройти до нижніх дихальних шляхів і доставити туди ліки. В роботі використаний ультразвуковий генератор туману з напругою живлення 24В, потужністю 20Вт, з вбудованим датчиком рівня води.

3.6. Вибір датчика вологості

Розглянемо кожен тип датчика вологості в контексті даної задачі.

Для поточної задачі підходить ємнісний датчик вологості. Він має ряд переваг, зокрема низька вартість, лінійність характеристики і малий розмір. До того ж він має широкий діапазон вимірювання, який охоплює можливі значення кімнатної вологості. Недоліком можна вважати необхідність наявності генератора частоти, від якого залежить точність вимірювання.

Резистивний датчик також цілком задовольняє вимоги задачі внаслідок його значної точності і малих габаритів. Він здатний вимірювати вологість в широкому діапазоні величин, який достатній для використання в якості кімнатного гігрометра. Його перевагою для використання в проектованому приладі є порівняно проста схема вимірювання опору, який за допомогою процесора перетворюється у значення відносної вологості.

Отже, вимоги проектованого приладу задовольняють і конденсаторний, і резистивний датчики. Доцільно використати комбінований датчик вологості, який містить в датчик вологості та схему перетворення сигналу. З наявних на ринку був вибраний датчик DHT22. Це комбінований датчик вологості, який містить в собі і датчик вологості, і схему перетворення вихідного сигналу безпосередньо у значення вологості, і датчик температури, що дозволить вимірювати температуру повітря в кімнаті. Виміряні значення з датчика передаються по однопровідній лінії [17]. Їх зчитати можна з цифрового піна Arduino.

3.7. Вибір вентилятора

Ультразвуковий парогенератор здатний лише перетворити воду в пару, але за відсутності зовнішнього відбору створеної пари вона буде осідати назад у воду. Вентилятор варто вибрати без ШІМ і реалізувати її на стороні схеми управління, оскільки він буде працювати за умов підвищеної вологості, може контактувати з парою, а електронні компоненти бажано використовувати за вологості 40 – 60%. Доцільно використовувати вентилятор постійного струму з напругою живлення 24В. Максимальний розмір 40 на 40 мм, оскільки мінімального потоку повітря вистачить для відбору створеної пари. Але реально дуже складно знайти вентилятор такого малого розміру і з напругою живлення 24В, тому виберемо з напругою живлення 12В. Вибраний вентилятор Sunon EB40101S2-000U-999. Він має напругу живлення 12В і потужність 1,08Вт, що цілком задовольняє задачу [18].

3.8. Висновок до розділу 3

В даному розділі описана технологія ШІМ для регулювання потужності навантаження і її реалізація в Arduino. В проектованому приладі вона буде використовуватись для регулювання потужності ультразвукового випарювача.

На основі параметрів проектованого пристрою вибрана плата Arduino, яка мінімально задовольняє його вимоги. Вибраний дисплей для виводу інформації про параметри та режим роботи. Ним став двострічковий символьний дисплей з шістнадцятьма стовпцями символів. На основі вимог вибраний водостійкий датчик температури для вимірювання температури води, яка залита в прилад. Також вибраний ультразвуковий пристрій для випарювання води, резистивний датчик вологості зі схемою перетворення і вентилятор для відводу створеної пари.

4. Розробка принципової схеми і розробка коду керування приладом

4.1. Підключення датчика вологості

Як було сказано вище, передача даних датчиком DHT22 здійснюється по однопровідній лінії. Вихідним піном є другий контакт датчика. Його необхідно підключити до будь-якого цифрового або аналогового в режимі цифрового вивиду Arduino. Підключимо до дев'ятого цифрового контакту. Для взаємодії з датчиком, зокрема задання піну, куди підключений датчик, типу датчика, ініціалізації та зчитування інформації з нього необхідно підключити бібліотеку «DHT.h». Вона призначена для взаємодії з датчиками типу DHT [8].

Наступними етапами підключення датчика є налаштування відповідного піну в режим входу і ініціалізація, тобто, подання сигналу старту на датчик. Ці дії здійснюються в розділі setup():

```
pinMode(PIN_DHT, INPUT);
dht.begin();
```

Відповідно функція pinMode() дозволяє призначити будь-який пін на вхід чи вихід, а dht.begin() запускає датчик вологості.

Далі необхідно зчитати температуру і вологість з датчика. Вперше це зробимо відразу після ініціалізації в розділі setup():

```
humidityCurrent = dht.readHumidity();
t = dht.readTemperature();
```

Наступні зчитування будуть проводитись в розділі loop() циклічно і це виконуватиме такий код:

```
if( (( millis() - readDelay) > 1000 ) ) {
   humidityCurrent = dht.readHumidity();
   t = dht.readTemperature();
   readDelay = millis();
```

```
}
```

Тут крім безпосередньо зчитування параметрів вологості і температури реалізована ще затримка тривалістю 1000 мс = 1 с між зчитуваннями. Це пов’язано з тим, що датчик DHT22 можна опитувати не частіше одного разу за секунду. Варто звернути увагу, що температура і вологість зчитуються з точністю до двох знаків після коми, але для індикації достатньо використати температуру з точністю до десятих, а значення вологості взяти з точністю до цілих.

4.2. Підключення датчика температури

Терморезистор PT100 не можна під’єднувати безпосередньо обома контактами до Arduino, оскільки така схема не спрацює, бо в платформі відсутні амперметри на входах і, відповідно, за законом Ома не вдасться визначити опір датчика. Тоді необхідно включити датчик за схемою дільника напруги в парі з постійним резистором і за допомогою аналогово-цифрового перетворювача (далі АЦП) визначати величину напруги в точці між резисторами. Постійний резистор бажано вибирати такого номіналу, який якнайближчий до опору датчика. Резистор з більшим опором зменшить точність вимірювань опору, оскільки чим більший його опір, тим менше порівняно з ним буде змінюватись опір датчика при зміні температури. Меншу зміну напруги АЦП Arduino може не розпізнати. Крім цього постійний резистор більшого опору має більший розкид номіналу в одиницях опору навіть з малим відсотком допуску відхилень. Таким чином резистор з опором в 1000 Ом і допуском 1% може мати відхилення опору до 10 Ом в більшу чи меншу сторону, а цього вже достатньо для спотворення вимірюваної напруги. Резистор зі значно меншим опором не бажано вибирати, оскільки тоді зменшиться падіння напруги на ньому, що спричинить збільшення величини струму, що протікає через послідовно з’єднані резистор і датчик. Для резистора величина струму не має ваги, але для датчика вона важлива,

оскільки надто великий струм буде нагрівати датчик зсередини і це також призведе до спотворень вимірювань. Виходячи з вище сказаного постійний резистор R3 в дільникові напруги виберемо рівним 100 Ом і з допуском 1%. Нижче одного відсотка допуск немає сенсу вибирати, оскільки такі резистори дорожчі і їх використання не виправдане для даної задачі. Так само не варто вибирати резистор з допуском більше одного відсотка, бо це уже помітно вплине на результати.

Найкращу точність вимірювань опору через дільник напруги можна отримати лише тоді, коли опір постійного резистора вимірюється точним вимірювальним приладом і отримана величина вписується в формулу для розрахунку опору.

В результаті маємо схему підключення термістора, яка наведена на рисунку 4.1. В нашому випадку вимірювання напруги в середній точці здійснюється через аналоговий вхід Arduino з номером 0. Для цього можна використати будь-який інший аналоговий вхід, але не цифровий, бо він може зчитувати лише значення логічного нуля і логічної одиниці.

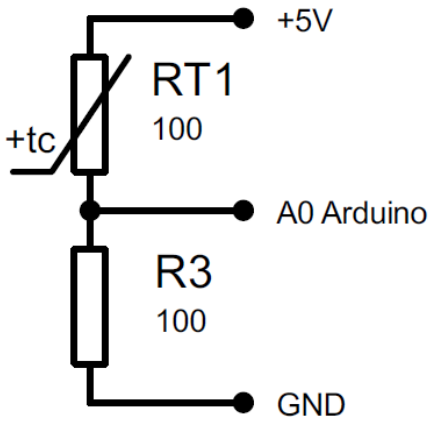


Рисунок 4.1. Схема підключення термістора до Arduino

Розрахуємо потужність, яку буде споживати датчик температури. За законом Ома розрахуємо струм через дільник напруги:

$$I = \frac{U}{RT1 + R3} = \frac{5}{100 + 100} = 0{,}025\ (\text{А}); \tag{4.1}$$

Підставимо отримане значення у формулу потужності:

$$P = I^2 R = (0{,}025)^2 \cdot 100 = 0{,}0625\ (\text{Вт}) = 62{,}5\ (\text{мВт}). \tag{4.2}$$

Такої потужності недостатньо, щоб помітно спотворити результати вимірювань, враховуючи при цьому ще доволі великі розміри датчика і те, що його корпус виготовлений зі сталі, яка має високу темплоємність. Тому вибрані рішення можна вважати вдалими.

Запишемо формулу для знаходження напруги на термісторові для даного дільника напруги:

$$U_{RT1} = U_{\text{оп}} \frac{R_{RT1}}{R_{RT1} + R_3}; \tag{4.3}$$

де U_{RT1} – напруга на термісторові; $U_{\text{оп}}$ – опорна напруга; R_{RT1}і R_3 – значення відповідних опорів. Виразимо опір R_{RT1}:

$$R_{RT1} = \frac{R_3}{\frac{U_{\text{оп}}}{U_{RT1}} - 1}; \tag{4.4}$$

За останньою формулою будемо знаходити опір датчика температури, але для початку необхідно визначитись зі значенням $U_{\text{оп}}$. В нашому випадку опорна напруга рівна напрузі живлення Arduino і дорівнює 5 В, але АЦП Arduino Nano працює не з вольтами, а з восьмибітним виглядом зчитуваної напруги. Тому $U_{\text{оп}}$ необхідно взяти рівним 1023 – максимальне значення, яке може зчитати АЦП [8].

Далі підключення датчика реалізуємо на рівні програмного коду. В цьому випадку підключати додаткові бібліотеки непотрібно. Для початку для зручності роботи назвемо вхід A0 коді програми назвою, яка виражає його призначення, а саме PIN_WATER_TEMP і зробимо це таким кодом на початку програми:

```
#define PIN_WATER_TEMP A0
```

Наступним кодом налаштуємо аналоговий контакт A0 на режим входу:

```
pinMode(PIN_WATER_TEMP, INPUT);
```

Далі необхідно зчитати напругу в середній точці і перевести її в опір:

```
thermistorResistance=constantResistor*(1023.0/analogRead(PIN_WATER_TEMP)-1);
```

де thermistorResistance – змінна, в яку записується розрахований опір; constantResistor – номінал резистора в другому плечі дільника напруги; analogRead(pin) – функція, яка зчитує значення аналогової напруги в діапазоні від 0 до 1023 на вході pin та повертає зчитане значення.

Далі перетворимо визначимо температуру через опір таким кодом:

```
waterTempCurrent = (thermistorResistance - defaultThermistorResistance)/
ohmByDegrees;
```

де waterTempCurrent – змінна, в яку записується розрахована температура; defaultThermistorResistance – опір терморезистора при 0 °C; ohmByDegrees – температурний коефіцієнт терморезистора.

На цьому підключення терморезистора завершено, розрахована температура далі будемо зчитувати зі змінної waterTempCurrent.

4.3. Підключення кнопок

Керування приладом буде здійснюватися за допомогою двох кнопок. Щоб уникнути додавання ще двох кнопок для підтвердження входу в режим та виходу з нього будемо розрізняти коротке та довге натискання кнопки. Це дозволить на кожній кнопці отримати ще по одній варіації, а сумарно матимемо чотири. Кнопки назвемо «+» або «вгору» та «–» або «вниз». Коротке натискання кнопки «+» буде збільшувати значення, яке можна змінювати в поточному меню або перемикатиме пункти меню, якщо це головне меню, аналогічно коротке натискання кнопки «–» зменшуватиме змінюване значення або перемикатиме пункти меню, якщо це головне меню. Довге натискання кнопки «+» призведе до переходу у вибраний пункт меню або до підтвердження змін і переходу в наступне меню залежно від того, в якому розділі меню вона була натиснута. Довге натискання кнопки «–» в будь-якому розділі меню призводить до повернення в головне меню і зупинки випаровувача та вентилятора.

Серед різноманіття кнопок була вибрана тактова кнопка, оскільки за допомогою неї не будуть комутуватись високі струми і напруги, до того ж цей вид кнопок дуже поширений. Конкретна реалізація для наскрізного чи поверхневого монтажу повинна бути вибрана уже при проектуванні корпусу, але бажано вибирати для наскрізного монтажу, оскільки такі кнопки краще закріплені на платі, тому краще перенесуть часті натискання.

Кнопки підключимо до аналогових входів A4 та A5, які працюють в режимі цифрових. За відсутності будь-якого сигналу на аналоговому вході у випадку зчитування з нього значення будемо випадково отримувати значення логічних нуля або одиниці, тому для стабільності роботи до входів Arduino необхідно підключити резистори номіналом порядку 1-10 кОм, які іншим виводом підключені на землю. Резистори більшого номіналу можуть не забезпечити необхідну стабільність, а меншого призведуть до значного енергоспоживання при натисканні.

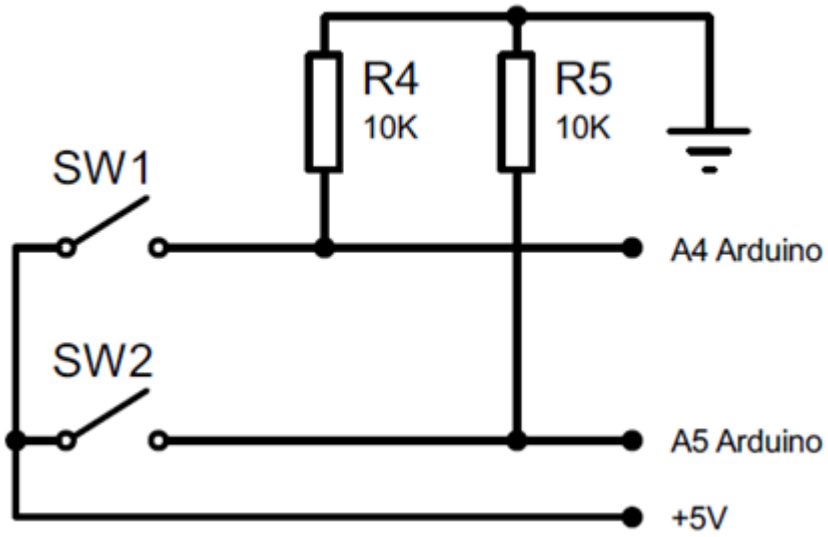


Рисунок 4.2. Схема підключення кнопок до Arduino

Виберемо резистор номіналом 10 кОм. Цього буде достатньо, щоб забезпечити на входах чіткий рівень логічного нуля. При натисканні кнопок через відповідний резистор буде протікати струм величиною 0,5 мА. На основі вище наведених даних розроблена схема підключення кнопок до Arduino. Вона наведена на рисунку 4.2. Код для роботи з кнопками наведений нижче:

```
if (digitalRead(PIN_BUTTON_UP) == HIGH && buttonFlagUp == 0) {
   timeCounterUp = millis();     buttonFlagUp = 1;
   clickDelay = millis();
}
if ((millis() - clickDelay) > 50) {
   if (digitalRead(PIN_BUTTON_UP) == LOW && buttonFlagUp == 1) {
      unsigned long timeIntervalUp = millis() - timeCounterUp;
      buttonFlagUp = 0;
      if (timeIntervalUp < timeInterval) {
         keyPressedUpShort = true;
      } else {
         keyPressedUpLong = true;
      }
   }
}
```

Він призначений для обробки натискань кнопки «+». Структурно він складається з двох частин: зчитування натискання та відпускання кнопки. Для уникнення брязкоту контактів і, відповідно, помилкових спрацювань, додана затримка обробки натискань тривалістю 50 мс. Цей код також ділить натискання кнопки на довге та коротке, вираховуючи чам між моментом натискання і відпускання. Код обробки натискань кнопки «-» структурно аналогічний.

4.4. Робота з дисплеєм

4.4.1. Фізичне підключення дисплею

Вивід інформації буде здійснюватися на символьний дисплей RAYSTAR RC1602B. Призначення виводів цього дисплею [15] наведено в таблиці 4.1.

Таблиця 4.1. Призначення контактів RAYSTAR RC1602B

Номер контакту	Позначення	Опис
1	Vss	Мінус живлення логіки
2	Vdd	Плюс живлення логіки
3	Vo	Регулювання контрасту
4	RS	Сигнал вибору даних/інструкцій
5	RW	Сигнал вибору режиму читання/запису
6	E	Сигнал ввімкнення
7	DB0	Шина даних
8	DB1	Шина даних
9	DB2	Шина даних
10	DB3	Шина даних
11	DB4	Шина даних
12	DB5	Шина даних
13	DB6	Шина даних
14	DB7	Шина даних
15	A	Плюс живлення підсвітки
16	K	Мінус живлення підсвітки

З таблиці 4.1 видно, що живлення дисплею здійснюється через перші два контакти. Даний дисплей може бути підключений до джерела напруги 3 В або 5 В. Підключимо його до того ж джерела живлення, що і Arduino. Також до землі підключимо 16 контакт, а підсвітку дисплею будем вмикати через 15 контакт підключивши його до Arduino через резистор 220 Ом. На схемі він позначений R6. Для регулювання контрасту використаємо змінний резистор опором 10 кОм. Два крайні виводи підключимо до землі і джерела живлення 5 В, а середню точку до контакту Vo дисплею. Налаштування контрасту необхідно проводити уже після збору пристрою з включеною підсвіткою, бо з вимкненою підсвіткою контраст може бути добрим, а з її включенням значно погіршати внаслідок освітлення пікселів. Критерієм хорошого контрасту є добре помітний виведений на екран текст і непомітні інші пікселі, які вимкнені. Контакт R/W необхідно підключити на землю, оскільки в даному

пристрої ми не використовуємо функцію читання з дисплею, а лише записуємо в нього.

Даний дисплей може працювати в двох режимах: 8-бітному і 4-бітному. Для роботи в 8-бітному режимі використовуються старші і молодші DB0-DB7, а в 4-бітному лише молодші DB4-DB7 біти. 8-бітний режим дозволяє вдвічі частіше оновлювати інформацію на екрані, ніж 4-бітний, бо в останньому передача даних та інструкцій здійснюється за два такти: спочатку старші 4 біти, потім молодші. Максимальна кількість оновлень даних за секунду для цього дисплею становить близько десяти. Тому в 4-бітному режимі отримаємо близько п’яти оновлень, цього цілком достатньо для індикації і режиму, і параметрів, оскільки в проектованому пристрої не передбачено індикацію величин, які швидко змінюються в часі. До того ж 4-бітний режим потребує використання чотирьох контактів контролера для передачі даних та інструкцій. Контакти RS та E підключаються до цифрових пінів [15]. Отже, підключимо дисплей до Arduino таким чином: RS до D7, E до D6, DB4 – DB7 до D5 – D2 відповідно, підсвітку A до D12.

4.4.2. Програмне підключення дисплею і виведення інформації

Далі налаштуємо роботу Arduino з дисплеєм. Для цього підключимо стандартну бібліотеку для роботи з дисплеями “LiquidCrystal.h”. Вона містить функції для переміщення курсора на дисплеї, виведення даних, створення символів, очищення дисплею тощо. Наведемо деякі з них, які використані в коді програми:

- lcd.clear() – функція очищує екран дисплею перед новим виведенням інформації, при написанні коду виявилось, що її слід використовувати в таких частинах коду, які виконуються лише за певних подій, наприклад, після натискання кнопок, інакше отримаєм мерехтіння на екрані внаслідок постійного стирання і запису нової інформації;

- lcd.setCursor(x, y) – встановлює дисплей в точку з координатами x та y, для даного дисплею x може набувати значень від 0 до 15, а y може бути рівним лише 0 та 1;
- lcd.print() – виводить на дисплей текст чи дані, які знаходяться в дужках.

Далі підключимо дисплей наступною стрічкою коду [8]:

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

Таким чином ми задали програмно задали до яких цифрових контактів підключені виводи RS, E, DB4 – DB7. Останнім кроком є ініціалізація дисплею командою lcd.begin(16, 2) в розділі setup(). Далі за допомогою вище описаних функцій можна виводити дані на екран.

4.4.3. Створення символу градуса

При роботі пристрою нам необхідно виводити значок градуса на дисплей, але цей значок відсутній в даному дисплеї, тому його необхідно створити. Пам’ять дисплею дозволяє записати в нього до восьми додаткових символів. Кожен символ відображається на матриці 5x8 точок і для створення символу градуса необхідно задати які точки треба засвітити. В таблиці 4.2 підсвічено сірим значок, який необхідно отримати і записані відповідні стани пікселів.

Таблиця 4.2. Стан пікселів для показу символу градуса

0	0	1	1	1
0	0	1	0	1
0	0	1	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Далі необхідно записати бітову маску символу, який створюємо і отриману маску присвоїти змінній бітового типу. Код, що це здійснює наступний:

```
byte tempCel [8] =
{
 0b00111,
 0b00101,
 0b00111,
 0b00000,
 0b00000,
 0b00000,
 0b00000,
 0b00000
};
```

Створений символ запишемо в пам’ять дисплею кодом:

```
lcd.createChar(1, tempCel);
```

В цьому запис цифра 1 означає номер, під яким символ записаний в пам’ять, можливі значенні від 1 до 7. Тепер символ створений і його можна відобразити написанням «\1».

4.4.4. Налаштування підсвічування

Контакт підсвічування дисплею можна підключити до джерела живлення, але тоді вона буде використовувати енергію весь час і привертатиме до себе увагу, тому нею необхідно керувати. Для цього необхідно розробити відповідний код, структурно нам необхідно реалізувати ввімкнення підсвічування при натисканні будь-якої кнопки, вимкнення за відсутності дій протягом заданого часу, продовження світіння на наступний інтервал часу при виконанні будь-яких дій.

Вмикає підсвічування наступний код:

```
 if ((keyPressedUpShort || keyPressedUpLong || keyPressedDownShort ||
keyPressedDownLong) && !lightEnable) {
    digitalWrite(PIN_LCD_LIGHT, 1);
    lightingTimer = millis();
    keyPressedUpLong = false;
    keyPressedUpShort = false;
    keyPressedDownLong = false;
    keyPressedDownShort = false;
    lightEnable = true;
  }
```

Він спрацьовує за умови, що натиснута будь-яка кнопка з будь-якою тривалістю натискання і при цьому підсвічування було вимкнено. Далі на контакт підсвічування записується логічна одиниця, записується у відповідну змінну момент виконання даної функції, віджимаються усі кнопки і записується в змінну lightEnable стан підсвічування, в даному випадку true. Таким чином перше натискання не спричинить ніяких дій в програмі, а лише спричинить підсвічування.

Якщо заданий час відсутні натискання кнопок, то наступний код вимкне підсвічування:

```
 if ( ((millis() - lightingTimer)/1000) > durationOfLighting) {
    digitalWrite(PIN_LCD_LIGHT, 0);
    lightEnable = false;
  }
```

Він визначає різницю часу між моментом ввімкнення підсвічування і поточною миттю. Тривалість підсвічування можна задати записавши потрібний час в секундах у змінну durationOfLighting. Далі, аналогічно до попередньо блоку коду, на вихід контакту підсвічування записується логічний нуль і в змінну, яка відповідає за стан підсвічування, значення false.

Для уникнення незручностей при використанні пристрою необхідно подовжувати час свічення при виконанні будь-яких дій:

```
 if ((keyPressedUpShort || keyPressedUpLong || keyPressedDownShort || keyPressedDownLong)) {
    lightingTimer = millis();
 }
```

Наведений вище код буде записувати в змінну lightingTimer новий час, відносно якого буде визначатись момент вимкнення підсвічування.

4.5. Підключення і керування випаровувачем та вентилятором

Для керування вентилятором і випаровувачем використовуємо ключі на польових транзисторах. Біполярні не використовуємо, оскільки вони споживають більше потужності, порівняно з польовими, і таким чином більше навантажують порти Arduino по струму, для даної задачі вони не мають переваг. Польовий транзистор необхідно вибрати такий, щоб він витримував напругу стік-витік 24В, зі струмом стік-витік не менше 1А, керуючою напругою 5 В. Виберемо n-канальний IRF520. Він має наступні характеристики: максимальна напруга стік-витік 100В, максимальний струм стік-витік 6,5А, максимальна напруга затвору 20В, розсіювана потужність до 50Вт [19]. Він має кращі характеристики, ніж бажані для задачі, але його використання виправдане низькою вартість і поширеністю. Для спрощення схеми такі транзистори використаємо як ключі і для вентилятора, і для випаровувача.

Затвори транзисторів підключаються безпосередньо до цифрових виходів Arduino. Затвори транзисторів мають певну ємність, яка впливає на їх роботу і на роботу інших компонентів схеми. В даному випадку зарядження і розрядження їх буде відбуватись через порт Arduino, вплинути на зарядження ми не можем, але можем зменшити навантаження при розрядженні. Для цього затвор кожного транзистора через резистор 1кОм подамо на землю. Такий

номінал вибраний, щоб мінімально навантажувати мікросхему при ввімкненні високого рівня на виході порту і при цьому щоб забезпечити швидке розрядження ємності, оскільки величина ємності досить мала.

Транзистори позначимо Q1 для керування випаровувачем, Q2 для керування вентилятором. Q1 підключаємо до цифрового контакту D11, Q2 до D10. В коді програми відповідні контакти прописуються як виходи:

```
pinMode(PIN_OUT_EVAP, OUTPUT);
digitalWrite(PIN_OUT_EVAP, 0);
pinMode(PIN_OUT_COOLER, OUTPUT);
digitalWrite(PIN_OUT_COOLER, 0);
```

Вище в перших двох стрічках задаємо контакт D11 як вихід і подаємо на вихід логічний 0, інакше на виході буде випадкове значення. Аналогічно в наступних двох стрічках робимо з D10. Далі управління вентилятором зводиться до його вмикання-вимикання за необхідності. Вмикання здійснюється таким кодом:

```
digitalWrite(PIN_OUT_COOLER, 1);
```

Вимикання здійснюється аналогічно, тільки замість логічної одиниці записується логічний нуль.

Дещо інакше керуємо випаровувачем, оскільки необхідно регулювати його потужність роботи за допомогою ШІМ. В поточному приладі керування потужністю випаровувача, а конкретно інтенсивністю створення пари, здійснюється наступним блоком коду:

```
steamIntensityPWM = map(steamIntensity, 0, 100, 0, 255);
analogWrite(PIN_OUT_EVAP, steamIntensityPWM);
```

Далі наведена розшифровка використаних змінних і функцій:

- steamIntensity – змінна, яка містить задану потужність випаровування в діапазоні від 1 до 100%.
- map(value, a1, a2, b1, b2) – функція для перетворення числа з діапазону від a1 до a2 в число діапазону від b1 до b2. В даному випадку значення

змінної steamIntensity з діапазону 0 – 100 перетворюється в число з діапазону 0 – 255, який використовується в ШІМ.

- PIN_OUT_EVAP – вихід, з якого здійснюється подача ШІМ сигналу на навантаження.
- steamIntensityPWM – змінна, яка зберігає значення, яке необхідно виставити на виході за допомогою ШІМ.

4.6. Розробка меню приладу і коду керування

Для зручності керування приладом розроблено меню з мінімальною кількістю налаштувань і максимальною інформативністю. Його блок-схема наведена на рисунку (рис. 4.3). В головному меню (рис. 4.4) за допомогою коротких натискань кнопок можна переміщувати курсор на необхідний пункт. Підтвердження вибору здійснюється натисканням тривалістю довше 1 секунди.

В режимі зволоження (рис. 4.5) на екрані відображається поточна вологість після тексту “hC=”, бажана вологість після тексту “hF=” і поточна температура повітря в градусах Цельсію після “t=”. Бажану вологість можна збільшувати і зменшувати з кроком 2% коротким натисканням відповідних кнопок, в цьому випадку зміна здійснюється в реальному часі. Вийти з цього режиму, як і з будь-якого іншого можна тривалим натисненням кнопки “вниз”.

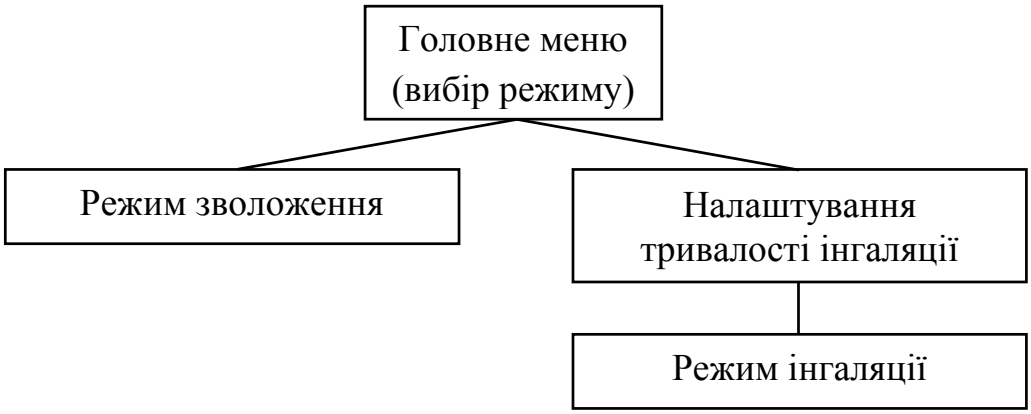


Рисунок 4.3. Блок-схема меню

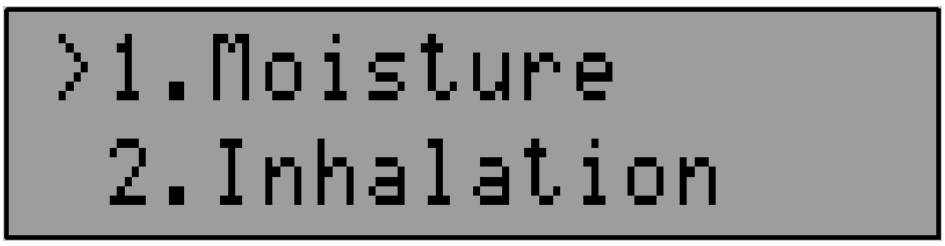


Рисунок 4.4. Головне меню приладу

В меню налаштування тривалості інгаляції (рис. 4.6) можна виставити бажану тривалість інгаляції з кроком і мінімальною тривалістю 1 хвилина. Підтвердити час можна тривалим натисненням кнопки “вгору”.

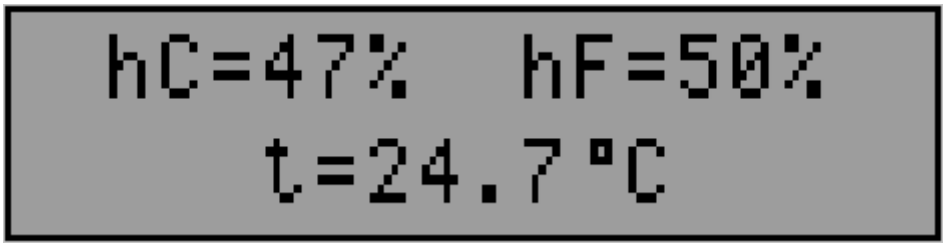


Рисунок. 4.5. Меню зволоження

Рисунок. 4.6. Меню налаштування тривалості інгаляції

Після цього відбудеться перехід у наступне меню (рис. 4.7), відповідно до блок-схеми, і початок інгаляції. В цьому режимі на екрані показано час від

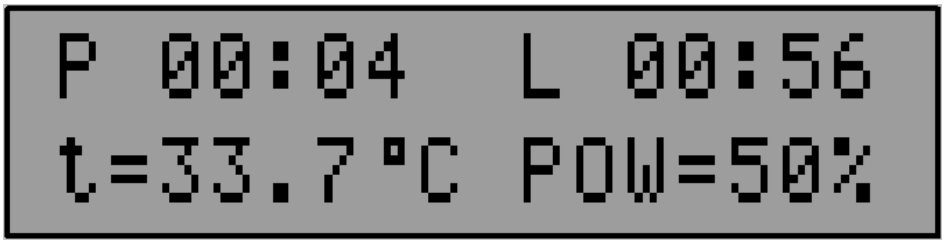


Рисунок. 4.7. Режим інгаляції

початку інгаляції після букви “P”, час до закінчення після букви “L”, поточна температура води після “t=” і поточна потужність роботи випаровувача у відсотках після тексту “POW=”. Останній параметр можна змінювати коротким натисканням кнопок з кроком 5% в діапазоні від 0 до 100%.

Рисунок. 4.8. Сповіщення про закінчення інгаляції

Після закінчення виставленого часу на екран буде виведено інформацію про закінчення інгаляції (рис. 4.8). Також в режимі інгаляції відбувається контроль температури води. При перевищенні максимальної чи зниженні нижче мінімальної межі інгаляція зупиняється і на екран виводиться попередження про вихід за допустимі межі. Відповідні зображення екрану наведені на рисунках 4.9 та 4.10.

Рисунок. 4.9. Попередження про надто холодну воду

Рисунок. 4.10. Попередження про надто гарячу воду

4.7. Розробка схеми живлення приладу

В розроблюваній схемі є компоненти з трьома напругами живлення: 5В, 12В і 24В. Для роботи кожного з компонентів потрібно на них подати саме таку напруги з мінімальним відхиленням, інакше компоненти будуть працювати нестабільно або вийдуть з ладу. Тому необхідно розробити такий блок живлення, який забезпечить на виході усі три напруги.

Є такі варіанти реалізації живлення схеми: блок живлення з однією вихідною напругою в комбінації з цифровими підвищувачами або понижувачами напруги, блок живлення, який видаватиме на виході усі три напруги.

Розглянемо блок живлення з трьома вихідними напругами. Такий блок живлення має трансформатор і схему перетворення змінного струму в постійний. А оскільки в трансформатора три вторинні обмотки з різними вихідними напругами, то, відповідно, потрібно три фільтри. Це зробить схему дуже громіздкою.

Блок живлення з однією вихідною напругою в комбінації з цифровими підвищувачами теж має місце для використання в проектованому приладі. Його доцільно використовувати, якщо пристрій, який живиться підвищеною напругою має малу потужність, інакше будуть, порівняно, значні втрати на нагрівання. В нашому випадку найменшу потужність споживає цифрова частина схеми з напругою живлення 5В, трохи більше вентилятор 12В і найбільше випаровувач з напругою живлення 24В. Виходячи з описаних вище причин такий блок живлення використовувати недоцільно.

Блок живлення з однією вихідною напругою в комбінації з цифровими понижувачами напруги найдоцільніше використовувати для живлення пристрою, що розробляється. Для цього необхідно використати блок живлення з вихідною напругою 24В.

Живлення випарювача буде здійснюватись безпосередньо від блока живлення. Для живлення вентилятора з двигуном 12В необхідно понизити

напругу за допомогою цифрового понижувача постійної напруги. Для цього використаємо мікросхему L7812. Вона з двома конденсаторами зовнішньої обв’язки реалізує перетворювач постійної напруги з виходом 12В. Дана мікросхема має три виводи: Vin, Vout, GND – вхід, вихід і земля відповідно. Вхідна напруга можу бути в межах від 15 до 35 В, в нашому випадку вона буде рівна 24В, отже, мікросхема працюватиме з допустимим запасом по характеристиках. Вихідний струм може бути до 1,5А, чого вистачить для живлення цифрової частини схеми і невеликого вентилятора. На випадок перевантаження в мікросхему вбудовані схеми захисту від перегріву і перевищення максимального вихідного струму, які вимкнуть вихід до повернення нормального режиму роботи. Вентилятор споживає струм 0,08А, тоді наступний перетворювач напруги і цифрова частина схеми можуть спожити до 1,42А [20].

Цифрова частина має напругу живлення 5В, тому для її живлення необхідно використати ще один цифровий понижувач постійної напруги з вихідною напругою 5В. Виберемо з тої ж лінійки мікросхему L7805. Вона має усі аналогічні характеристики і схеми захисту . Також в неї аналогічна схема включення і обв’язка [20].

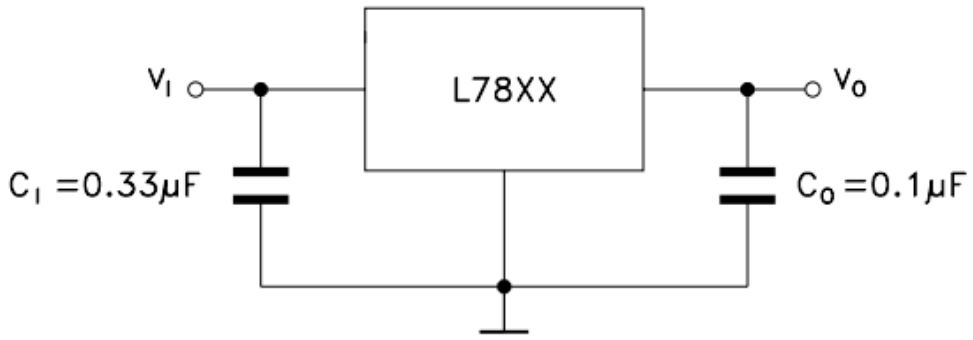


Рис.4.11. Схема включення мікросхем лінійки L78XX [20]

Дли живлення схеми необхідно використовувати блок живлення з вихідною напругою 24В і потужністю не менше 25 Вт.

4.8. Висновок до розділу 4

В поточному розділі описані принципи підключення усіх використовуваних компонентів до Arduino, зокрема наведені схеми їх фізичного підключення і код, який дозволяє працювати з відповідними компонентами. Була створена блок-схема меню дисплею і результат її реалізації. На основі вище наведених даних розроблена принципова схема приладу, яка наведена в додатку 1 і написаний код, який необхідно записати в Arduino. Лістинг коду наведений в додатку 2. Схема була відтворена в середовищі симуляції електронних схем Proteus Design 8, код програми скомпільований і завантажений в симульовану плату Arduino Nano. Тестування схеми пройшло успішно, тому її можна відтворювати на реальних компонентах і при правильному відтворенні схеми вона запрацює з першого разу.

5. Висновки

В дипломні роботі було здійснено огляд можливих реалізацій кожного блоку приладу і вибрано найкращі, виходячи з призначення і умов використання кожного компоненту. Важливим критерієм при підборі компонентів було мінімальне задоволення вимог до використання. Це важливо для ефективного користування ресурсами. Для керування потужністю навантаження використана ШІМ. Наведена інформація щодо реалізації цього методу в Arduino і синтаксис функції, яка дозволяє ввімкнути ШІМ на конкретному виході контролера. На основі вибраних компонентів розроблена принципова схема приладу, яка призначена до підключення до джерела постійного струму з напругою 24В. Наведені вимоги до блоку живлення, який необхідний для роботи даного приладу.

Розроблена схема приладу відтворена в Proteus Design 8. При симуляції усі помічені недоліки коду були виправлені. Оскільки схема успішно просимульована, то її можна відтворювати уже на реальних компонентах.

Перелік використаної літератури

1. Perry, R.H. and Green, D.W., (2007) Perry's Chemical Engineers' Handbook (8th Edition), Section 12-3, Psychrometry, McGraw-Hill. URL: https://www.academia.edu/34738909/Perrys_Chemical_Engineers_Handbook_8th Ed (дата доступу 20.05.2019).

2. "Climate - Humidity indexes". Encyclopaedia Britannica. URL: https://www.britannica.com/science/climate-meteorology/Humidity-indexes (дата доступу 17.05.2019).

3. "Climate/humidity table". Transport Information Service of the German Insurance Association. URL: http://www.tis-gdv.de/tis_e/misc/klima.htm/ (дата доступу 17.05.2019).

4. "Optimum Humidity Levels for Home". URL: https://www.airbetter.org/optimum-humidity-levels-home/ (дата доступу 18.05.2019).

5. Effect of Humidity and Condensation on Power Electronics Systems. URL:https://www.semikron.com/dl/service-support/downloads/download/ semikron-application-note-effect-of-humidity-and-condensation-on-power-electronics-systems-en-2016-07-15-rev-00/ (дата доступу 20.05.2019).

6. А.Г.Чучалин, Н.П.Княжеская, М.О.Потапова Место небулайзеров в ингаляционной терапии хронических обструктивных заболеваний легких. // Русский медицинский журнал. – 2006. – №7– С. 521–524.

7. Arduino – Products. URL: https://www.arduino.cc/en/Main/Products (дата доступу 25.05.2019).

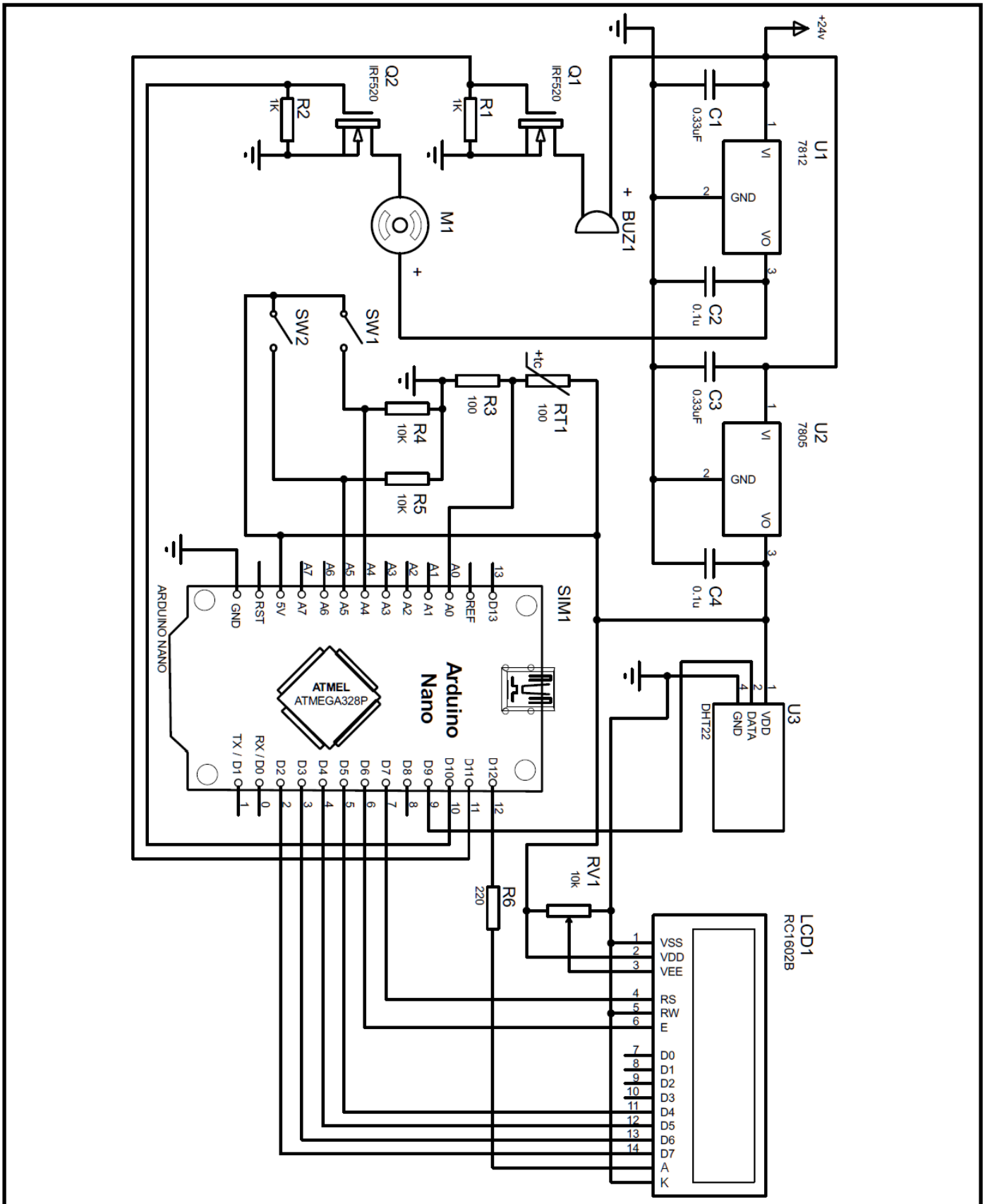
8. Петин В.А. Проекты с использованием контроллера Arduino. – Санкт-Петербург, 2014. с. 20 – 66.

9. Г. Виглеб. Датчики. Устройство и применение. – Москва: «Мир», 1989. с. 14 – 120.

10. А. Г. Дивин, С. В. Пономарев. Методы и средства измерений, испытаний и контроля. Часть 3. – Тамбов: ФГБОУ ВПО «ТГТУ», 2013. с. 31 – 32.

11. Umidificatore a caldo Respirasano. URL: http://www.chicco.by/manuals/staroe/User_Manual_Healthy_Breathing.pdf (дата доступу 26.05.2019).
12. Ultrasonic Humidifiers. URL: https://www.cedengineering.com/userfiles/Ultrasonic%20Humidifiers.pdf (дата доступу 26.05.2019).
13. Ивченко Ю.А., Федоров А.А. Чем измерить влажность? // Датчики и системы. – 2003. - №8. – С. 53 – 54.
14. Analog Pulse Width Modulation. Texas Instruments. URL: http://www.ti.com/lit/ug/slau508/slau508.pdf (дата доступу 27.05.2019).
15. RC1602B-datasheet. URL: https://www.raystar-optronics.com/upload_files/monochrome-lcd-module/16x2-character-lcd-display/RC1602B-datasheet.pdf (дата доступу 27.05.2019).
16. Mini Thermocouple (Thermal Resistance). PT100 Datasheet. URL: http://www.wr-wz.com/common/down/name/580d5472a1a32.pdf.html (дата доступу 28.05.2019).
17. Digital-output relative humidity & temperature sensor/module DHT22 (DHT22 also named as AM2302). URL: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf (дата доступу 28.05.2019).
18. EB40101S2-000U-999 Datasheet. URL: https://www.jameco.com/Jameco/Products/ProdDS/2167234.pdf (дата доступу 29.05.2019).
19. Power MOSFET IRF520, SiHF520 Datasheet. URL: https://www.vishay.com/docs/91017/91017.pdf (дата доступу 29.05.2019).
20. Positive voltage regulators L7800 series. URL: https://www.jameco.com/Jameco/Products/ProdDS/889305.pdf (дата доступу 30.05.2019).

Додаток 1.



Змн.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Юрчик Ю.І.			Принципова схема	Літ. / Арк. 54 / Акрушів 57
Перевір.		Кутова О.Ю.				
Реценз.						НТУУ «КПІ ім. І.Сікорського» ФЕЛ
Н. Контр.						
Затверд.						

Додаток 2.

Познач.	Найменування	Кіл.	Примітка
	Конденсатори		
C1, C3	0,33 мкФ 50 В	2	
C2, C4	0,1 мкФ 50 В	2	
	Резистори		
R1, R2	1 кОм 5% 0,125 Вт	2	
R3	100 Ом 1% 0,125 Вт	1	
R4, R5	10 кОм 5% 0,125 Вт	2	
R6	220 Ом 5% 0,125 Вт	1	
RV1	10 кОм 5% 0,125 Вт	1	
	Датчики		
RT1	PT100	1	термодатчик
U3	DHT22	1	датчик вологості
	Дисплеї		
LCD1	RC1602B	1	
	Транзистори		
Q1, Q2	IRF520	2	
	Модулі		
SIM1	Arduino Nano	1	
	Двигуни		
M1	EB40101S2-000U-999		вентилятор
	Випаровувачі		
BUZ1	AW1602 24В 20Вт	1	
	Мікросхеми		
U1	L7812	1	
U2	L7805	1	

Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Юрчик Ю.І.			Перелік компонентів	Літ.	Арк. 55 / Аркушів 57
Перевір.		Кутова О.Ю.					
Реценз.						НТУУ «КПІ ім. І.Сікорського» ФЕЛ	
Н. Контр.							
Затверд.							

Додаток 3. Код програми

```
#include <DHT.h>
#include <DHT.h>
#include <LiquidCrystal.h>

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

#define PIN_DHT 9 // пін для сигналу, який поступає з датчика
#define PIN_OUT_EVAP 11
#define PIN_OUT_COOLER 10
#define PIN_LCD_LIGHT 12
#define PIN_WATER_TEMP A0
#define PIN_BUTTON_UP A4
#define PIN_BUTTON_DOWN A5
#define DHT_TYPE DHT22 // DHT 22

int menu = 22;
// menu = 0 - головне меню
// menu = 11 - режим зволоження
// menu = 21 - режим інгаляції, налаштування
// menu = 22 - режим інгаляції, індикація

int curs = 0;
int cursDel = 1;
int buttonFlagUp = 0;
int buttonFlagDown = 0;
int inhalationTimePassedSec = 0;
int inhalationTimeLeftSec = 0;
int passedMin = 0;
int passedSec = 0;
int leftMin = 0;
int leftSec = 0;
int steamIntensityPWM;
// межа часу між коротким і довгим натисканням кнопки
int timeInterval = 150;
// затримка натискання (мс)
```

```
36:   int delayClick = 10;
37:   // тривалість підсвітки екрану (с)
38:   int durationOfLighting = 10;
39:   // стандартна вологість у %
40:   int humidityFinal = 50;
41:   // стандартна тривалість інгаляції у хв
42:   int inhalationDuration = 1;
43:   // інтенсивність випаровування при інгаляції у %
44:   int steamIntensity = 50;
45:   // максимальна температура води при інгаляції
46:   int waterTempMax = 50;
47:   // мінімальна температура води при інгаляції
48:   int waterTempMin = 25;
49:
50:   // температурний коефіцієнт опору термодатчика
51:   float ohmByDegrees = 0.39;
52:   float waterTempCurrent = 0;
53:   float thermistorResistance = 0;
54:   float lightingTimer = 0;
55:   float humidityCurrent;
56:   float t;
57:
58:   unsigned long timeCounterUp;
59:   unsigned long timeCounterDown;
60:   unsigned long timeStartInh;
61:   unsigned long clickDuration;
62:   unsigned long readDelay;
63:
64:   boolean keyPressedUpLong = false;
65:   boolean keyPressedUpShort = false;
66:   boolean keyPressedDownLong = false;
67:   boolean keyPressedDownShort = false;
68:   boolean lightEnable = true;
69:
70:   byte tempCel [8] =
71:   {
72:     0b00111,
```

```
73:     0b00101,
74:     0b00111,
75:     0b00000,
76:     0b00000,
77:     0b00000,
78:     0b00000,
79:     0b00000
80:   };
81:
82:   // ініціалізація датчика вологості
83:   DHT dht(PIN_DHT, DHT_TYPE);
84:
85:   void setup() {
86:     pinMode(PIN_OUT_EVAP, OUTPUT);
87:     digitalWrite(PIN_OUT_EVAP, 0);
88:
89:     pinMode(PIN_OUT_COOLER, OUTPUT);
90:     digitalWrite(PIN_OUT_COOLER, 0);
91:
92:     pinMode(PIN_LCD_LIGHT, OUTPUT);
93:     digitalWrite(PIN_LCD_LIGHT, 1);
94:
95:     pinMode(PIN_DHT, INPUT);
96:     pinMode(PIN_WATER_TEMP, INPUT);
97:     pinMode(PIN_BUTTON_UP, INPUT);
98:     pinMode(PIN_BUTTON_DOWN, INPUT);
99:     // запуск дисплею
100:          lcd.begin(16, 2);
101:          // створення значка градуса
102:          lcd.createChar(1, tempCel);
103:          // запускаєм датчик вологості DHT22
104:          dht.begin();
105:
106:          humidityCurrent = dht.readHumidity();
107:          t = dht.readTemperature();
108:          }
109:
```

```
110:        void loop() {
111:          if( (( millis() - readDelay) > 1000 ) ) {
112:            humidityCurrent = dht.readHumidity();
113:            t = dht.readTemperature();
114:            readDelay = millis();
115:          }
116:          // визначення короткого і довгого натискання кнопки ВВЕРХ
117:          if (digitalRead(PIN_BUTTON_UP) == HIGH && buttonFlagUp == 0) {
118:            // записав час першого натискання у змінну
119:            timeCounterUp = millis();
120:            // записав індикатор першого натискання
121:            buttonFlagUp = 1;
122:            clickDuration = millis();
123:          }
124:          if ((millis() - clickDuration) > delayClick) {
125:            if (digitalRead(PIN_BUTTON_UP) == LOW && buttonFlagUp == 1) {
126:              unsigned long timeIntervalUp = millis() - timeCounterUp;
127:              // обнулив індикатор першого натискання
128:              buttonFlagUp = 0;
129:
130:              if (timeIntervalUp < timeInterval) {
131:                keyPressedUpShort = true;
132:              } else {
133:                keyPressedUpLong = true;
134:              }
135:            }
136:          }
137:          // кінець визначення короткого і довгого натискання кнопки
    ВВЕРХ
138:          // визначення короткого і довгого натискання кнопки ВНИЗ
139:          if (digitalRead(PIN_BUTTON_DOWN) == HIGH && buttonFlagDown
    == 0) {
140:            // записав час першого натискання у змінну
141:            timeCounterDown = millis();
142:            // записав індикатор першого натискання
143:            buttonFlagDown = 1;
144:            clickDuration = millis();
```

```
145:          }
146:          if ((millis() - clickDuration) > delayClick) {
147:            if (digitalRead(PIN_BUTTON_DOWN) == LOW && buttonFlagDown
   == 1) {
148:              unsigned long timeIntervalDown = millis() - timeCounterDown;
149:              // обнулив індикатор першого натискання
150:              buttonFlagDown = 0;
151:
152:              if (timeIntervalDown < timeInterval) {
153:                keyPressedDownShort = true;
154:              } else {
155:                keyPressedDownLong = true;
156:              }
157:            }
158:          }
159:          // кінець визначення короткого і довгого натискання кнопки ВНИЗ
160:          // код керування підсвіткою
161:          if ((keyPressedUpShort || keyPressedUpLong ||
   keyPressedDownShort || keyPressedDownLong) && !lightEnable) {
162:            digitalWrite(PIN_LCD_LIGHT, 1);
163:            lightingTimer = millis();
164:            keyPressedUpLong = false;
165:            keyPressedUpShort = false;
166:            keyPressedDownLong = false;
167:            keyPressedDownShort = false;
168:            lightEnable = true;
169:          }
170:          if ((keyPressedUpShort || keyPressedUpLong ||
   keyPressedDownShort || keyPressedDownLong)) {
171:            lightingTimer = millis();
172:          }
173:          if ( ((millis() - lightingTimer)/1000) > durationOfLighting ) {
174:            digitalWrite(PIN_LCD_LIGHT, 0);
175:            lightEnable = false;
176:          }
177:          // кінець коду керування підсвіткою
178:          // вибір пункту в головному меню
```

```
179:          if (menu == 0 && keyPressedUpLong == true) {
180:            keyPressedUpLong = false;
181:            lcd.clear();
182:            if (curs == 0) {
183:              menu = 11;
184:            } else {
185:              menu = 21;
186:            }
187:          }
188:          // кінець вибору пункта в головному меню
189:          // код налаштування часу інгаляції і перехід в меню інгаляції
190:          if (menu == 21 && keyPressedUpLong == true) {
191:            keyPressedUpLong = false;
192:            lcd.clear();
193:            menu = 22;
194:          }
195:          // кінець коду налаштування часу інгаляції і переходу в меню
    інгаляції
196:          // код повернення в головне меню з підменю
197:          if (keyPressedDownLong == true) {
198:            keyPressedDownLong = false;
199:            lcd.clear();
200:            digitalWrite(PIN_OUT_EVAP, 0);
201:            digitalWrite(PIN_OUT_COOLER, 0);
202:            menu = 0;
203:            timeStartInh = 0;
204:          }
205:          // кінець коду повернення в головне меню з підменю
206:          switch (menu) {
207:            case 0:    // головне меню
208:            {
209:              // визначення куди треба перемістити курсор в залежності від
    натискання кнопки
210:              if ((keyPressedUpShort == true) || (keyPressedDownShort ==
    true)) {
211:                keyPressedUpShort = false;
212:                keyPressedDownShort = false;
```

```
213:                if (curs == 0) {
214:                   curs = 1;
215:                   cursDel = 0;
216:                } else {
217:                   curs = 0;
218:                   cursDel = 1;
219:                }
220:             }
221:             // кінець визначення куди треба перемістити курсор в
    залежності від натискання кнопки
222:             // вивід курсора меню
223:             lcd.setCursor(0, curs);
224:             lcd.print(">");
225:             lcd.setCursor(0, cursDel);
226:             lcd.print(" ");
227:             // кінець виводу курсора меню
228:             // вивід пунктів меню
229:             lcd.setCursor(1, 0);
230:             lcd.print("1.Moisture");
231:             lcd.setCursor(1, 1);
232:             lcd.print("2.Inhalation");
233:             // кінець виводу пунктів меню
234:             break;
235:          }
236:          case 11:    // режим зволоження
237:          {
238:             // вивід поточної вологості
239:             lcd.setCursor(1, 0);
240:             lcd.print("hC=");
241:             lcd.print(humidityCurrent, 0);
242:             lcd.print("%  ");
243:             // кінець виводу поточної вологості
244:             // вивід бажаної вологості
245:             lcd.setCursor(9, 0);
246:             lcd.print("hF=");
247:             lcd.print(humidityFinal);
248:             lcd.print("%  ");
```

```
249:              // кінець виводу бажаної вологості
250:              // вивід поточної температури повітря
251:              lcd.setCursor(4, 1);
252:              lcd.print("t=");
253:              lcd.print(t,1);
254:              lcd.print("\1C   ");
255:              // кінець виводу поточної температури повітря
256:              // ввімкнення та вимкнення випаровувача і вентилятора
257:              if (humidityCurrent <= humidityFinal) {
258:                 digitalWrite(PIN_OUT_EVAP, 1);
259:                 digitalWrite(PIN_OUT_COOLER, 1);
260:              } else {
261:                 digitalWrite(PIN_OUT_EVAP, 0);
262:                 digitalWrite(PIN_OUT_COOLER, 0);
263:              }
264:              // кінець ввімкнення та вимкнення випаровувача і
   вентилятора
265:              // збільшення та зменшення рівня бажаної вологості
266:              if (keyPressedUpShort == true && humidityFinal < 100) {
267:                 keyPressedUpShort = false;
268:                 humidityFinal += 2;
269:              }
270:              if (keyPressedDownShort && humidityFinal > 0) {
271:                 keyPressedDownShort = false;
272:                 humidityFinal -= 2;
273:              }
274:              // кінець збільшення та зменшення рівня бажаної вологості
275:              break;
276:           }
277:           case 21:    // режим інгаляції, налаштування
278:           {
279:              // збільшення та зменшення тривалості інгаляції
280:              if (keyPressedUpShort == true && inhalationDuration < 30) {
281:                 keyPressedUpShort = false;
282:                 inhalationDuration += 1;
283:              }
284:              if (keyPressedDownShort && inhalationDuration > 1) {
```

```
285:                keyPressedDownShort = false;
286:                inhalationDuration -= 1;
287:             }
288:             // кінець збільшення та зменшення тривалості інгаляції
289:             // вивід бажаної тривалості інгаляції на екран
290:             lcd.setCursor(3, 0);
291:             lcd.print("Duration:");
292:             lcd.setCursor(5, 1);
293:             lcd.print(inhalationDuration);
294:             lcd.print(" min ");
295:             // кінець виводу бажаної тривалості інгаляції на екран
296:             break;
297:          }
298:          case 22:    // режим інгаляції, індикація
299:          {
300:             if (timeStartInh == 0) {
301:                timeStartInh = millis();
302:             }
303:             inhalationTimePassedSec = (millis()-timeStartInh)/1000;
304:             // перевірка чи не закінчився час інгаляції
305:             if ( inhalationTimePassedSec < inhalationDuration*60 ) {
306:                // зчитування температури води
307:                thermistorResistance =
    100*(1023.0/analogRead(PIN_WATER_TEMP)-1);
308:                waterTempCurrent = (thermistorResistance -
    100)/ohmByDegrees;
309:
310:                // кінець зчитування температури води
311:                if (waterTempCurrent < waterTempMin || waterTempCurrent
    > waterTempMax) {
312:                   // перевірка температури води
313:                   if ( waterTempCurrent < waterTempMin ) {
314:                      lcd.setCursor(0, 0);
315:                      lcd.print("Too cold water");
316:                   }
317:                   if ( waterTempCurrent > waterTempMax ) {
318:                      lcd.setCursor(0, 0);
```

```
319:                     lcd.print("Too hot water");
320:                   }
321:                   digitalWrite(PIN_OUT_EVAP, 0);
322:                   digitalWrite(PIN_OUT_COOLER, 0);
323:                   // вивід поточної температури води
324:                   lcd.setCursor(0, 1);
325:                   lcd.print("t=");
326:                   lcd.print(waterTempCurrent);
327:                   lcd.print("\1");
328:                   // кінець виводу поточної температури води
329:                 } else {
330:                   passedMin = inhalationTimePassedSec/60;
331:                   passedSec = inhalationTimePassedSec - passedMin*60;
332:                   // вивід часу з моменту початку інгаляції
333:                   if (passedMin < 10) {
334:                     lcd.setCursor(0, 0);
335:                     lcd.print("P ");
336:                     lcd.print("0");
337:                     lcd.print(passedMin);
338:                   } else {
339:                     lcd.setCursor(2, 0);
340:                     lcd.print(passedMin);
341:                   }
342:                     lcd.print(":");
343:                   if (passedSec < 10) {
344:                     lcd.setCursor(5, 0);
345:                     lcd.print("0");
346:                     lcd.print(passedSec);
347:                   } else {
348:                     lcd.setCursor(5, 0);
349:                     lcd.print(passedSec);
350:                   }
351:                     lcd.print("  ");
352:                   // кінець виводу часу з моменту початку інгаляції
353:                   inhalationTimeLeftSec = inhalationDuration*60 -
    inhalationTimePassedSec;
354:                   leftMin = inhalationTimeLeftSec/60;
```

```
355:                leftSec = inhalationTimeLeftSec - leftMin*60;
356:                // вивід часу до кінця інгаляції
357:                if (leftMin < 10) {
358:                  lcd.setCursor(9, 0);
359:                  lcd.print("L ");
360:                  lcd.print("0");
361:                  lcd.print(leftMin);
362:                  lcd.print(":");
363:                } else {
364:                  lcd.setCursor(11, 0);
365:                  lcd.print(leftMin);
366:                }
367:                if (leftSec < 10) {
368:                  lcd.setCursor(14, 0);
369:                  lcd.print("0");
370:                  lcd.print(leftSec);
371:                } else {
372:                  lcd.setCursor(14, 0);
373:                  lcd.print(leftSec);
374:                }
375:                // кінець виводу часу до кінця інгаляції
376:                // вивід поточної температури води
377:                lcd.setCursor(0, 1);
378:                lcd.print("t=");
379:                lcd.print(waterTempCurrent,1);
380:                lcd.print("\1 ");
381:                // кінець виводу поточної температури води
382:                // задання інтенсивності випаровування
383:                if (keyPressedUpShort == true && steamIntensity < 100) {
384:                  keyPressedUpShort = false;
385:                  steamIntensity += 5;
386:                }
387:                if (keyPressedDownShort && steamIntensity > 0) {
388:                  keyPressedDownShort = false;
389:                  steamIntensity -= 5;
390:                }
391:                // кінець задання інтенсивності випаровування
```

```
392:                    // вивід поточної інтенсивності випаровування
393:                    lcd.setCursor(9, 1);
394:                    lcd.print("POW=");
395:                    lcd.print(steamIntensity,1);
396:                    lcd.print("%  ");
397:                    // кінець виводу поточної інтенсивності випаровування
398:                    // керування інтенсивністю випаровування
399:                    steamIntensityPWM = map(steamIntensity, 0, 100, 0, 255);
400:                    analogWrite(PIN_OUT_EVAP, steamIntensityPWM);
401:                    digitalWrite(PIN_OUT_COOLER, 1);
402:                    // кінець керування інтенсивністю випаровування
403:                  }
404:                } else {
405:                  lcd.setCursor(0, 0);
406:                  lcd.print("   Inhalation   ");
407:                  lcd.setCursor(0, 1);
408:                  lcd.print("    is over     ");
409:                  digitalWrite(PIN_OUT_EVAP, 0);
410:                  digitalWrite(PIN_OUT_COOLER, 0);
411:                }
412:                break;
413:              }
414:            } // закриття switch
415:            keyPressedUpLong = false;
416:            keyPressedUpShort = false;
417:            keyPressedDownLong = false;
418:            keyPressedDownShort = false;
419:          } // закриття loop
```