МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

МІЖНАРОДНИЙ КЛАСИЧНИЙ УНІВЕРСИТЕТ

ІМЕНІ ПИЛИПА ОРЛИКА

Економічно-технологічний факультет

Кафедра інженерних технологій

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського рівня) вищої освіти

на тему:

«Розробка автоматизованої системи моніторингу параметрів мікроклімату на базі платформи Arduino»

Зі спеціальності 123

«КОМП’ЮТЕРНА ІНЖЕНЕРІЯ»

Виконавець:

Захаров В.О.

Науковий керівник:

к.т.н., доц. Гайша О.О.

Миколаїв – 2025

ЗМІСТ

Перелік умовних скорочень 4
Вступ 5
Розділ 1 Аналіз проблеми підтримки параметрів мікроклімату вир+ичих приміщень на заданому рівні 8
1.1 Аналіз типів виробничої діяльності, що потребують стабільного мікроклімату, та відповідні вимоги до нього 8
1.2 Структура та основні характеристики автоматизованої системи моніторингу мікроклімату. Постановка задачі 12
1.3 Висновки по розділу 16
Розділ 2. Розробка апаратної частини системи моніторингу мікроклімату виробничих приміщень 18
2.1. Обґрунтування вибору мікроконтроллеру, як основи для зведення системи моніторингу мікроклімату 18
2.2. Вибір типу та схеми розміщення датчиків системи моніторингу мікроклімату 25
2.3. Комунікаційні властивості проектованої системи. 29
2.4 Висновки по розділу 36
Розділ 3. Розробка програмного забезпечення системи моніторингу мікроклімату виробничих приміщень 38
3.1. Вибір типу та особливостей інтерфейсу користувача проектованої системи 38
3.2. Вибір інструментальних засобів для розробки 41
3.2.1. Обґрунтування вибору технології програмування 41
3.2.2. Вибір мови програмування 48
3.2.3. Вибір середовища розробки та додаткових програмних засобів 53

3.3. Формалізація алгоритмічної складової програмно-апаратного комплексу, що розробляється 54
3.4. Особливості програмної реалізації системи моніторингу мікроклімату виробничих приміщень 56
3.5. Документаційне забезпечення розробленого програмно-апаратного комплексу 58
3.5.1. Інструкція про розгортання апаратної частини системи 58
3.5.2. Інструкція адміністратора створеної системи 59
3.5.3. Інструкція користувача розробленої системи 60
3.6. Тестування та випробування розробленої системи моніторингу мікроклімату 61
3.7. Висновки по розділу 63
Висновки 65
Перелік використаних джерел 66
Додаток А. Електрична принципова схема системи. 68
Додаток Б. Висхідні коди розробленого програмного забезпечення. 69

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CPU	Central Processor Unit
IDE	Integrated Development Environment
PC	Personal Computer
PWM	Pulse Wideness Modulation
UI	User Interface
БД	База даних
ІТ	Інформаційні технології
ОО	Об’єктно-орієнтований (-на, -не)
ООП	Об’єктно-орієнтоване програмування
ОС	Операційна система
ПЗ	Програмне забезпечення
ПК	Персональний комп’ютер
САУ	Система автоматичного управління
СУБД	Система управління базами даних
ШІМ	Широтно-імпульсна модуляція

ВСТУП

За час розвитку науки і техніки людина навчилася більш-менш ефективно вирішувати задачі налаштування параметрів навколишнього середовища під власні потреби. Спочатку це були засоби обігріву приміщень природним паливом (елементарно: дровами), пізніше – центральним опаленням, електрикою. У кінці XX ст. широкого поширення отримали і засоби кондиціонування приміщень, що дозволили ефективно боротися з іншою стороною «медалі» – високою температурою. Таким чином, із впевненістю можна констатувати, що на сьогоднішній день людина може ефективно контролювати температуру у приміщеннях, де мешкає.

У той же час задачі забезпечення температурних режимів у приміщеннях іншого призначення, зокрема, виробничих чи складських, можуть становити не меншу, а навіть більшу актуальність. Дійсно, якщо зниження температури житлових приміщень влітку забезпечує лише комфорт (і тільки в окремих випадках є дійсно потрібним, як, наприклад, у місцях утримання серцево-судинних хворих), то, наприклад, для збереження м’ясо-молочної продукції воно є просто життєво необхідним.

Якщо говорити про виробничі процеси, то часто, окрім певного діапазону температур, критичними умовами нормального їх протікання є також певні допустимі значення вологості повітря, чистоти повітря робочої зони, тощо. Забезпечення хімічної (від сторонніх речовин), механічної (від пилу), біологічної (від вірусів та бактерій) чистоти повітря, а також інших контактних для виробничого процесу середовищ, є надзвичайно специфічною задачею, яку слід вирішувати інженерними превентивними методами, однак малоймовірним є використання для цих цілей якихось активних очищаючих пристроїв, що працювали би в автоматизованому, чи автоматичному режимах під управлінням системи автоматичного управління (далі – САУ). Складність зведення САУ для контролю чистоти середовищ у великій мірі також обумовлена відсутністю надійних датчиків, що могли би визначати результуючі вихідні параметри такої

системи (отже, неможливою є побудова контурів обернених зв'язків та організації автоматизованого управління в цілому). У будь-якому випадку, відносна кількість задач, що потребують чисельного контролю чистоти середовищ є набагато меншою, ніж проблем забезпечення необхідних температур (а також і вологості повітря), тому в подальшій роботі ці задачі розглядатися не будуть.

Що ж стосується вологості, то цей параметр хоча і є у деякій мірі менш важливим, ніж температура, але у значній кількості виробничих (технологічних) процесів, все ж вимоги до неї виписані досить докладно. Зважаючи на те, що забезпечити контроль вологості повітря додатково до контролю температури не є надзвичайно складною задачею, в подальшому будемо говорити саме про дві цих величини: температуру і вологість (відносну) повітря. Їх комбінацію коротко будемо називати у даній роботі «мікрокліматом», і питання забезпечення двох згаданих параметрів мікроклімату на бажаному рівні є дуже **актуальним** у великій кількості виробничих процесів.

Об'єктом дослідження є процес моніторингу значень заданих параметрів мікроклімату виробничих приміщень.

Предметом дослідження є комп'ютеризовані системи (їх апаратні та програмні складові), що можуть забезпечувати ефективний моніторинг параметрів мікроклімату виробничих приміщень.

Зважаючи на проблему, описану вище, можна сформувати наступну **мету дослідження**: для убезпечення виробничого процесу слід підвищити ефективність інформування оператора (або виконуючого блоку САУ) про поточні значення параметрів мікроклімату виробничого приміщення, зокрема, забезпечити термінову доставку відповідної тривожної інформації до уповноважених осіб при виході параметрів мікроклімату за дозволені межі. Для досягнення мети слід вирішити наступні **задачі дослідження**:

- проаналізувати проблему забезпечення параметрів мікроклімату приміщень на заданому рівні та обґрунтувати необхідність розробки;
- здійснити аналіз необхідних складових автоматизованої системи моніторингу параметрів мікроклімату;

- обрати елементи апаратної частини проектованої системи, розробити її структуру та обрати основні параметри;

- здійснити розробку програмної складової проектованої системи;

- виконати тестування системи та створити необхідну документацію.

В роботі застосовуються наступні **методи дослідження**: загальнонаукові методи аналізу та синтезу, спеціалізовані методи електроніки та програмування.

Новизна даного дослідження полягає в адаптації дешевої модульної системи розробки електронних пристроїв Arduino до вирішення реальних промислових задач.

Практична значущість роботи є досить високою, оскільки розроблене у ній рішення із мінімальними змінами (що враховують специфіку кожного конкретного об’єкту-приміщення) може бути застосоване на реальних підприємствах, убезпечуючи їх від «тихої» відмови обладнання (такого, як холодильне), яка часто приводить до великих збитків через порчу товарів за рахунок впливу високої температури.

В **перспективі** роботу можна розвинути, додаючи до розробленої системи моніторингу (тобто чистого спостереження) активні виконуючі блоки, що реалізовували б обернені зв’язки по температурі та вологості (наприклад, при виході температури за визначені межі – вмикали би резервний охолоджувач чи нагрівач, і т.п.). В такому випадку терміновість оперативного втручання людини можна було би значно зменшити, що покращувало б експлуатаційні характеристики системи і планується до виконання у подальшій роботі.

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ ПІДТРИМКИ ПАРАМЕТРІВ МІКРОКЛІМАТУ ВИР+ИЧИХ ПРИМІЩЕНЬ НА ЗАДАНОМУ РІВНІ

1.1 Аналіз типів виробничої діяльності, що потребують стабільного мікроклімату, та відповідні вимоги до нього

Як було зазначено у вступі, поняття «мікроклімат» є надзвичайно широким і може трактуватися у досить широких межах, залежно від конкретної предметної галузі, до якої це поняття застосовується. Однак, конкретно у даній роботі під мікрокліматом будемо розуміти поняття, що описується множиною C із двох елементів:

$$C = \{T, \varphi\}, \tag{1.1}$$

де $c_1 = T$ – температура, °C;

$c_2 = \varphi$ – відносна вологість повітря, %.

Розглянемо, які ж компанії або виробництва взагалі потребують контролю мікроклімату, що необхідно для встановлення меж обраних параметрів та вимог до їх точності, а втім – вибору конкретних датчиків, що могли б забезпечувати вимірювання із потрібними характеристиками.

В першу чергу, до підприємств, що потребують контролю температури, відносяться усі особи (юридичні та фізичні), що працюють із продовольством. Сюди відносяться як приміщення, де активно ведеться виробничий процес (переробка, обробка, безпосередні перетворення товарів продовольчої групи), так і складські площі, де відбувається зберігання (можливо із певними елементами технологічного процесу, як, наприклад, достигання лікеро-горілчаної продукції, яка повністю готова до використання, але додаткова часова витримка при певних умовах суттєво покращує її якість, а, отже, і вартісні показники; аналогічно – з сирокопченими та сиров’яленими ковбасами). При цьому більш актуальним є моніторинг мікроклімату у таких приміщеннях, де присутність людини є мінімальною, або взагалі не потрібна (крім, власне, контролю за параметрами

самого приміщення). Дійсно, якщо кондиціонер зламається, наприклад, на лінії розбирання рибної продукції (досить критична до температури ділянка), то працівники одразу це помітять і терміново сигналізуватимуть керівництву, або іншим, відповідальним за технічне забезпечення, особам. Незалежно від того, наскільки швидко при цьому буде полагоджено систему охолодження, у працівників принаймні буде час для дій, наприклад, для перенесення усіх критичних вантажів у інше, кондиціоноване/охолоджуване місце. В той же час, якщо ввечері після завершення робочого дня вийде з ладу промисловий холодильник у цеху достигання ковбасної продукції, то, за умови високої температури навколишнього середовища, тобто на вулиці (наприклад, через літній період, тропічні широти і т.д.), на ранок наступного дня (коли проблему вперше помітять працівники) уся продукція вже може бути зіпсованою. Для уникнення подібних ситуацій і створюється система моніторингу параметрів мікроклімату, яка при виході цих параметрів за дозволені межі, здійснюватиме ефективну тривогу. Про виникнення проблем із системами забезпечення температури та вологості, одразу буде повідомлено відповідальних осіб, які зможуть запровадити необхідних заходів.

Якщо говорити про конкретні температурні діапазони, то усі системи охолодження продовольчої продукції можна умовно розбити на два класи: холодильної та морозильної техніки. У холодильних камерах температура зазвичай підтримується на рівні 3-8 °С, причому в середній частині об’єму зазвичай значення є меншими (3-4 °С), а біля стінок підвищується (7-8 °С). Морозильні камери призначені для зміни агрегатного стану продуктів, які всі без виключення наповнені водою (навіть в’ялені вироби вміщують у собі невелику масову частку води): тією, що утворює цитоплазму клітин, а також міжклітинними рідинами. При зменшенні температури менше 0 °С (залежно від хімічного складу розчинів температура плавлення може знижуватися до величин порядку –1…–5 °С) відбувається кристалізація усієї води, що входить до складу продуктів і тим самим унеможливлюється деструктивна дія бактерій, які своєю дією поступово приводять продукти до непридатного стану. Таким чином, робочі

температури у морозильних камерах невеликого розміру складають величини порядку –6…–24 °С, однак при підтримці від'ємних температур у приміщеннях, абсолютне значення температури при цьому не може бути надто високим і зазвичай складає величину біля –6 °С.

Іноді для покращення консервативних властивостей виробничих приміщень використовують хімічні реагенти, як, наприклад, обробка газами при зберіганні бананів та деяких овочів, однак, по-перше, як уже було зазначено вище, у даній роботі розглядатиметься тільки контроль температури і вологості, а по-друге, у таких процесах все одно температура (а також і вологість) підлягають безперервному контролю, оскільки консервуючи дія газу є ефективною лише в комплексі з певними значеннями цих фізичних параметрів. Таким чином, і у цьому випадку система, що підлягає розробці у даній роботі, є необхідною та може бути взята за основу для процесу доробки та удосконалення.

Таким чином, широке застосування систем контролю температури наявне у галузі зберігання/переробки продовольчих товарів, а також – у галузі сільського господарства (овочесховища, сховища фруктів і т.п.). Слід відмітити, що підтримка низької температури у теплу пору року не є єдиним варіантом застосування таких систем. Дуже важливою є також і підтримка високої температури у холодну пору року у приміщеннях, де відбувається зростання/дозрівання рослинної продукції у штучних умовах, тобто у парниках. Дійсно, за умови порушення теплоізолюючих властивостей парника та зниження температури на період порядку доби можлива повна загибель врожаю із несенням 100%-х збитків відповідним підприємством. Отже, питання своєчасного встановлення несанкціонованого зниження температури у парнику є не менш важливим, ніж проблема детектування підвищення температур у сховищах сільськогосподарської продукції та готових продовольчих товарів. Відмітимо, що рекомендований температурний режим у парниках залежить від етапу, на якому знаходиться рослинна продукція всередині нього, але майже завжди ці температури повинні лежати у межах від 15 °С до приблизно 30 °С і складати розкид величиною 2-4 °С. Наприклад, до плодоносіння помідорів температура

має складати 19-22 °С, у період плодоносіння 26-30 °С, а після того, як знято перший врожай, температура має стабілізуватися на рівні 15-16 °С (для всіх випадків зберігаються вказані межі і розкид). Для інших типів рослин ситуація є схожою і вказані межі можна використовувати як орієнтири при виборі датчиків температури із певними характеристиками діапазону вимірювання та забезпечення точності.

Зростання рослин може потребувати і дещо інших умов мікроклімату, як, наприклад, на грибних підприємствах, де температури мають складати 22-25 °С при проростанні міцелію та 13-15 °С при плодоносінні грибів. Вологість повітря при цьому має становити біля 70 %.

Ще однією галуззю, що потребує контролю температури, є продаж живих квітів, адже і тут при підвищенні температури до 40 °С протягом цілої ночі загрожує зниженням естетичних характеристик квітів до неприйнятних значень (пожухлості, прим’ятості, пожовтіння і т.п.) та відповідними збитками для суб’єктів господарювання. Рекомендований діапазон температур складає від 0 до 10 °С для різних сортів квітів, тому при мішаному зберіганні (як у більшості магазинів) рекомендована температура складає біля 5 °С. Відмітимо, що життєвої необхідності для суспільства у збереженні великих обсягів квіткової продукції немає (на відміну від збереження в кондиційному стані великих обсягів їстівних продуктів), але в таких випадках доцільність контролю температури (та у меншій мірі – вологості, але на великих значеннях, а не малих – як при зберіганні продовольства) обумовлена потенційними фінансовими збитками окремих юридичних чи фізичних осіб, тобто в принципі також є обґрунтованою.

Серед галузей, що потребують стабілізації параметрів мікроклімату також можна виділити фармацевтику, адже переважна більшість лікарняних засобів є хімічними сполуками, що досить чутливі до температурних змін (а багато з них – і до вологості).

Таким чином, можна констатувати, що досить велика кількість видів виробничої діяльності потребує особливих умов мікроклімату (в основному температури та, у меншій мірі, вологості). Отже, розробка відповідних систем, що

виконували б функцію моніторингу параметрів мікроклімату в автоматичному (або напівавтоматичному) режимах являє собою надзвичайну актуальну промислову задачу, що має бути вирішена в рамках даного дослідження.

1.2 Структура та основні характеристики автоматизованої системи моніторингу мікроклімату. Постановка задачі

Проведений аналіз необхідних умов мікроклімату для різноманітних виробничих процесів показує, що діапазон температур, які можуть підлягати контролю, складає від –24 °С до +35 °С, а точність визначення температури для більшості випадків повинна складати біля одного градусу Цельсія:

$$T \in [-24; 35], {}^{\circ}C; \Delta T = \pm 1{}^{\circ}C . \qquad (1.2)$$

Умови (1.2) будуть пізніше покладені в основу вибору конкретних елементів апаратного забезпечення проектованої системи.

У питаннях відносної вологості повітря діапазон допустимих значень із звичайних природних обставин складає:

$$\varphi \in [0; 100], \%; \Delta\varphi = \pm 1\% ... \pm 5\% . \qquad (1.3)$$

Наведені умови (1.2)-(1.3) в подальшому будемо використовувати для вибору конкретних моделей датчиків та інших апаратних складових проектованої системи.

Окрім вимог до діапазонів контрольованих величин та точності їх визначення, важливими вхідними даними при побудові автоматизованої системи є відомості про її структуру та склад компонентів.

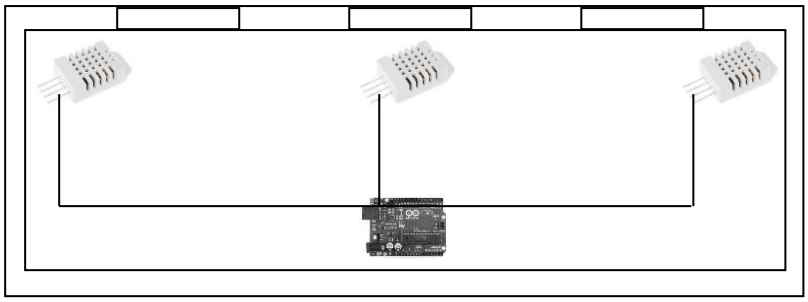
У випадку, що розглядається, можливою є побудова рішення на базі комп’ютерної системи, або мікроконтролерної. Перший варіант відрізняється значно більшою продуктивністю (як обчислювальною, так і інформаційною), а отже, і є більш гнучким. Однак, суттєвим недоліком використання універсальної комп’ютерної техніки для задач контролю одного-двох параметрів якоїсь системи чи об’єкта, а тим більше, простого їх моніторингу, є фінансова надмірність такого

рішення. Дійсно, для задач такого типу доцільним є використання мікроконтролерних пристроїв.

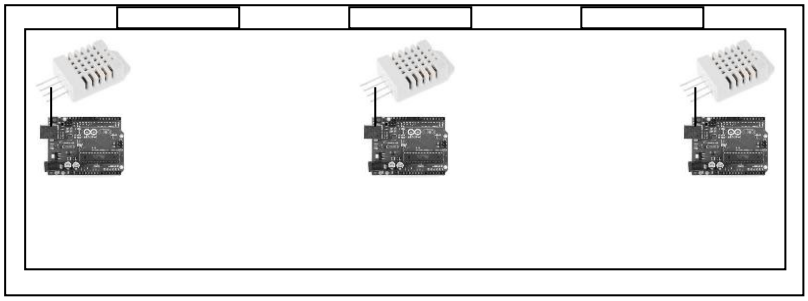
При використанні мікроконтролерів основними «органами чуттів» у них виступають датчики, що знімають з навколишнього середовища вхідні сигнали, на основі яких, власне мікроконтролером виробляються вихідні рішення по управлінню виконуючими пристроями, або, у випадку систем моніторингу, по простому інформуванню відповідальної особи про поточні параметри системи, за якою ведеться спостереження. Зазвичай, до одного мікроконтролера під'єднується декілька датчиків, максимальна кількість яких залежить від числа його входів. При цьому в цілому забезпечення стабільної температури у приміщенні передбачає розміщення декількох датчиків, кількість яких, взагалі кажучи, залежить від об'ємів цього приміщення (тобто і площі, і висоти). Однак, немає сенсу використовувати декілька датчиків температури, розміщених територіально поруч один з одним: для ефективного функціонування всієї системи, між ними має зберігатися певна, не дуже мала відстань.

Очевидно, при площі приміщення порядку 10 м^2 (і менше) потреби у кількох датчиках немає; реальна необхідність впровадження кількох датчиків виникає при площі контрольованого приміщення 20 м^2 і більше. Реальні холодильники на крупних птахофабриках досягають площ у сотні квадратних метрів, а у великих промислових теплицях площа приміщення взагалі може сягати тисячі квадратів при максимальних лінійних розмірах до 100 м (!). При таких розмірах робота системи з одним датчиком буде не дуже ефективною, оскільки нормальна температура з одного боку такого довгого приміщення зовсім не означає такої ж (або хоча б на мінімально прийнятному рівні) температури по усьому об'єму.

Таким чином, виникає питання про спосіб організації системи моніторингу мікроклімату, що має контролювати декілька точок виробничого приміщення в частині утворення єдиної мережі датчиків із центральним вузлом, або використання повністю незалежних пристроїв, розміщених у тих же точках, де у першому випадку розміщуються датчики. Обидва випадки показані на рис. 1.1.



а)



б)

Рис. 1.1. Два варіанти організації системи моніторингу: а – з одним контролером та комунікаційною мережею («зірка»); б – з окремими модульними пристроями без утворення мережі.

Кожен із двох способів організації системи моніторингу має свої плюси та мінуси, які зведемо у табл. 1.1.

Табл. 1.1. Порівняльний аналіз переваг та недоліків двох способів організації системи моніторингу.

Показник	Система з одним центральним контролером («зірка»)	Набір окремих пристроїв
Вартість	Потребує лише одного контролера, але значної кількості провідників. Цей варіант дещо дешевше (не набагато в абсолютному вираженні, оскільки вартість сучасних контролерів та	Провідники не потрібні, але кількість контролерів рівна кількості пристроїв. При використанні GSM-модулів також потрібно докупити їх та SIM-карти. Вартість цього варіанту у декілька разів вище

	модулів до них мала).	при кількості датчиків 2-3.
Надійність безвідмовної роботи системи	Низька, оскільки робота всієї системи залежить від одного центрального вузла.	Дуже висока, оскільки в системі є багатократне дублювання.
Ефективність системи	Еквівалентна при однаковій кількості датчиків та повній роботоспроможності системи.	
Складність розгортання системи	Висока, оскільки потребує монтажу кабелів, що завжди часозатратно.	Низька, окремі пристрої достатньо покласти на полиці у потрібних місцях.
Мобільність системи	Низька, потребує повного демонтажу системи.	Висока.
Уніфікованість системи	Середня, однаковими є лише провідники.	Висока (усі елементи однакові).
Масштабованість системи	Середня, потребує монтажу нового кабелю.	Дуже гарна, треба лише докласти нові пристрої у потрібні місця приміщення.

Як видно із таблиці, єдиною перевагою рішення типу «зірка» є його нижча вартість (причому, як мінімум, в рази), у порівнянні з другим варіантом, який має тотальне переважання з технічної точки зору. Однак, у даному випадку навіть не потрібно вирішувати задачу оптимізації, оскільки критерій вартості не є важливим. Справа в тому, що ціна сучасних електронних комплектуючих настільки мала, що навіть за умови використання 10 окремих пристроїв (тобто при переважанні вартості, припустимо, у 10 разів, хоча насправді, через вартість провідників зростання ціни не буде 10-кратним, а буде значно меншим), загальна вартість комплектуючих проекту буде складати біля 100 у.о., що становить надзвичайно мале значення для підприємств, що займаються торгівлею м’ясо-молочною продукцією і т.п. товарами.

Таким чином, однозначно (за усіма критеріями, крім вартості, яка не є суттєвою у даному випадку) слід обрати варіант організації системи із N окремих

однакових повністю функціональних пристроїв. Відповідно, у подальшій роботі розробці підлягає пристрій із зосередженими параметрами, що дозволяє вести моніторинг параметрів мікроклімату у даній конкретній точці виробничого приміщення.

Беручи до уваги вищенаведені відомості, можна сформувати наступну інформацію, що може бути прийнята у якості базової для розробки системи моніторингу мікроклімату виробничих приміщень (тобто виконати постановку задачі дослідження).

На основі вхідних даних про необхідні діапазони визначення температури (1.2) та вологості (1.3), а також про потрібну точність вимірювань (що задається тими ж співвідношеннями), а також беручи за основу концепцію використання повністю незалежних вимірювальних пристроїв (а не єдиної централізованої системи), необхідно розробити проект такого пристрою (програмно-апаратного рішення) для визначення параметрів мікроклімату, здійснити його реалізацію, провести тестування на предмет виконання поставленої задачі та зробити висновки про його ефективність та перспективи удосконалення.

1.3 Висновки по розділу

Таким чином, у розділі здійснено аналіз галузей, де існують нагальні потреби контролю параметрів мікроклімату, якими обрано температуру та відносну вологість повітря. Встановлено межі температурного діапазону, що підлягає вимірюванню у переважній більшості практичних задач (від –24 °C до +35 °C), а також вимоги до точності цього процесу (похибка не повинна складати більше одного градусу Цельсія). Відносна вологість повітря (відповідно до свого фізичного змісту) може змінюватися в обмежених границях від 0 до 100 %, а точність вимірювання повинна складати 1-2 %.

У розділі встановлено, що одного датчика температури для ефективної роботи системи у сучасних промислових приміщеннях може бути недостатньо

(при площах більше 20 м^2), і тоді виконано порівняльний аналіз двох варіантів організації системи із багатьма датчиками: централізований (типу «зірка») та у вигляді набору повністю незалежних пристроїв. Встановлено, що за усіма параметрами варіант із незалежними пристроями значно переважає централізований, крім, звичайно, ціни, але яка є несуттєвою в умовах даного дослідження. Відповідно, за базовий узято варіант із багатьма незалежними пристроями.

Керуючись цими даними можна переходити до процесу проектування апаратної, а потім і програмної частин системи, що розробляється.

РОЗДІЛ 2. РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ СИСТЕМИ МОНІТОРИНГУ МІКРОКЛІМАТУ ВИРОБНИЧИХ ПРИМІЩЕНЬ

2.1. Обґрунтування вибору мікроконтроллеру, як основи для зведення системи моніторингу мікроклімату

Основою для побудови будь-якої цифрової системи є її виконуючий пристрій – мікропроцесор – схема, що реалізує логіку роботи. Мікропроцесори можуть відрізнятися швидкістю роботи (робочою частотою, що на сьогодні вимірюється у сотнях і тисячах мегагерц), наявністю додаткових елементів у/на корпусі власне мікропроцесору. Так, уже традиційним рішенням, що значно підвищує продуктивність роботи центральних процесорів (Central Processor Unit – CPU) персональних комп'ютерів (ПК) є розміщення на його кристалі ще і блоку кеш-пам'яті (а часто – ще і кількох рівнів L1, L2, L3), яка раніше розміщувалася окремо на материнській платі ПК. Відмітимо, що у кеш-пам'яті відсутні ділянки даних чи коду, що зберігаються там на постійній основі. Кеш-пам'ять є зменшеним варіантом оперативної пам'яті, що має надзвичайно високу швидкість роботи. В той же час мікроконтролер також має в своєму складі кристал пам'яті, але зовсім іншого типу та призначення, а саме – флеш-пам'яті, яка повинна бути запрограмованою для довготривалого зберігання коду програми, що виконується мікроконтролером. Розмір флеш-пам'яті мікроконтролерів є досить малим і складає величину до кількох Мб, а часто – десятки кілобайтів. Програмний код процесора, як відомо, розміщується перед виконанням в оперативній пам'яті ПК, що має розміри порядку гігабайт, а зберігається на постійних запам'ятовуючих пристроях обсягами сотні гігабайт. Відповідно, вартість використання процесора (разом із усім, необхідним для його адекватної роботи додатковим обладнанням) для вирішення порівняно простих задач (таких, як моніторинг кількох фізичних величин у реальному світі) є необґрунтовано високою.

Більше того, саме призначення мікропроцесорів є набагато більш широким, аніж у контролерів. Особливість їх полягає у тому, що один і той же

мікропроцесор можна постійно використовувати для вирішення дуже і дуже різноманітних задач, запускаючи на ньому різне прикладне програмне забезпечення (далі – ПЗ). До речі, це прикладне ПЗ спирається на інший, специфічний рівень програмних продуктів – системне ПЗ, яким в першу чергу є операційна система (далі – ОС) ПК. Наявність ОС, що управляє апаратним забезпеченням комп'ютерної системи на найнижчому програмному рівні, є ще однією відмінністю мікропроцесора (що може використовуватися тільки з певною ОС), від мікро контролера. У мікроконтролер замість великої кількості прикладних програм прописаний один спеціалізований програмний продукт, який він весь час і виконує. Сама ця програма, прописана у флеш-пам'ять мікроконтролера може мати дуже і дуже різне функціональне призначення, вирішувати одну із дуже широкого кола прикладних задач, однак ця прикладна програма буде єдиною, що виконується мікроконтролером. Порівняння характеристик мікроконтролерів та мікропроцесорів виконано у вигляді табл. 2.1.

Табл. 2.1. Порівняння характеристик мікроконтролерів та мікропроцесорів.

Показник	Мікроконтролери	Мікропроцесори
Вартість	Дуже низька, порядку умовної одиниці.	Середня, порядку десятків у.о.
Надійність безвідмовної роботи системи	Дуже висока, пристрій при правильному монтажу та експлуатації практично не може вийти з ладу.	Висока, пристрій працює багато років.
Необхідність у додатковому обладнанні	Низька, потребує деяких дешевих компонентів типу тактового генератору, простих портів для підключення датчиків та зовнішніх пристроїв, і т.п.	Висока, потребує багато дорогих додаткових пристроїв (материнська плата, оперативна пам'ять, постійний запам'ятовуючий пристрій, і т.д.).

Універсальність системи	Дуже низька, призначений для виконання тільки однієї конкретної прикладної програми, адаптованої до апаратної обв'язки мікроконтролера.	Дуже висока, може виконувати будь-які прикладні програми, причому одночасно, в режимі розподілу процесорного часу.
Необхідність у додатковому програмному забезпеченні	Низька, потребує лише середовища розробки для створення управляючої програми та її трансферу до мікроконтролеру. В процесі роботи ніякого спеціалізованого додаткового програмного забезпечення не використовує.	Висока, в першу чергу потребує операційної системи (яка часто є достатньо дорогою), набору драйверів для можливості використання (або ефективної роботи) апаратного забезпечення, часто – набору проміжного ПЗ, як-то віртуальні машини, емулятори, і т.п.
Обчислювальна продуктивність	Мала, частоти порядку десятків мегагерц.	Висока, частоти порядку тисяч мегагерц.
Наявність власної постійної пам'яті	Так, вбудована флеш-пам'ять для збереження власної управляючої програми.	Ні, лише кеш-пам'ять та регістри, що аналогічні оперативній пам'яті.
Енергетична ефективність	Дуже висока, можуть роками працювати від одного акумулятора середньої ємності.	Середня, споживання у сотні і тисячі разів більше, ніж у мікроконтролерів (але при відповідно більшій продуктивності).
Порти введення-виведення	Наявні цифрові та часто аналогові однорозрядні порти низької продуктивності (порядку десятку-двох).	Наявні високопродуктивні шини та порти. Можливе їх розділення на окремі піни.

Аналізуючи табл. 2.1, можна зробити висновок, що для цілей даного дослідження однозначно (за усіма показниками) більш підходить мікроконтролер у зв’язку з наступними обставинами:

- система має лише одну функцію (призначення) – моніторинг мікроклімату;

- задача моніторингу (тим більше всього лише двох параметрів) не є високопродуктивною і не потребує значної кількості математичних обчислень, чи великих обсягів пам’яті;

- питання вартості не є дуже критичним, але при можливості вирішення проблеми малими фінансами, нелогічно і недоцільно залучати для цього додаткові кошти;

- бажано створити систему із якомога меншим енергоспоживанням, особливо, якщо взяти до уваги її високу мобільність, тобто небажаність прив’язки до точок підведення електроенергії, а забезпечення автономним живленням (до того ж невеликих малогабаритних показників).

Наступним кроком є вибір сімейства мікроконтролерів, і, незважаючи на суттєвий розвиток електронної галузі в останні десятиріччя, насправді реальний вибір тут є не дуже широким. На сьогоднішній день в реальних проектах застосовуються:

- мікроконтролери сімейства AVR компанії Atmel, що, ймовірно, є найбільш поширеними через цілий ряд обставин (в першу чергу, через велику інформаційну підтримку компанії та зацікавленого і мотивованого ком’юніті, значну кількість проектів, доступних у мережі Інтернет, величезну кількість додаткового обладнання, що гарно адаптується до використання разом із AVR, тощо);

- продукти під назвою PIC від компанії Microchip (менш поширені, як по кількості проектів, так і сумісного додаткового апаратного забезпечення; приблизно рівні по цінам відповідним продуктам AVR; мають вищий поріг входження в тему їх використання);

- пристрої ARM від однойменної фірми (які ще до сьогодні порівняно мало поширені, і в реальності знаходяться десь посередині між звичайними мікроконтролерами типу AVR/PIC та мікропроцесорами ПК: досить продуктивні,

але дешеві, універсальні, але такі, що мають порівняно просту архітектуру – ймовірно у майбутньому вони можуть замінити продукти AVR та PIC, але їх час просто ще не настав).

Аналізуючи особливості усіх трьох типів мікроконтролерів (та не розглядаючи інші екзотичні рішення), можна прийти до висновку, що для цілей даного дослідження найбільш прийнятним є сімейство AVR. Використання продукції саме цієї компанії має ще один величезний плюс, який полягає у грамотній маркетинговій політиці компанії, яка, розуміючи потреби широких мас молодих дослідників, випустила на базі своїх дуже популярних мікроконтролерів ATmega328 (та кількох аналогічних) цілу платформу Arduino. Спочатку ця концепція позиціонувалася як рішення для навчальної діяльності, оскільки у ній присутній цілий ряд особливостей, що ускладнюють процес застосування інших мікроконтролерів у реальних (працюючих) проектах.

Це, наприклад, встановлення на плату Arduino практично усього необхідного спектру мікропристроїв, які дозволяють працювати з нею у простих проектах без залучення якихось додаткових електронних елементів, тобто на Arduino вже наявна досить потужна так звана обв’язка: стабілізатор живлення, кварцовий резонатор, кола скидання, АЦП, порти цифрові та аналогові (цифрові, але такі, що імітують аналогові сигнали за допомогою широтно-імпульсної модуляції, далі – ШІМ або PWM – від Pulse Wideness Modulation). Ще однією перевагою всієї концепції є відсутність необхідності паяння, яке для більшості людей є складним процесом, а збір досить складних пристроїв за допомогою сполучних провідників (перемичок) на спеціальних макетних платах. На сьогоднішній день навіть готові (а не навчальні) системи на базі Arduino іноді продають на макетках (!). Також мікроконтролер Arduino не потребує спеціальних програматорів, а може записуватися за допомогою звичайного кабеля (від принтеру чи сканеру) типу USB Type-B. Ще одним фактором успіху всієї платформи стало зручне інтегроване середовище розробки (Integrated Development Environment – IDE) – Arduino Studio, в якому основним інструментом для створення програм («скетчів» - за термінологією Arduino) стала мова C/C++.

Для зовсім юних дослідників доступним є також досить зручний різновид мови Scratch, що називається Ardublock – рис. 2.1 (незважаючи на початкову направленість цієї блочної системи програмування на підлітків, для простих задач її часто використовують і цілком професійні електронники).

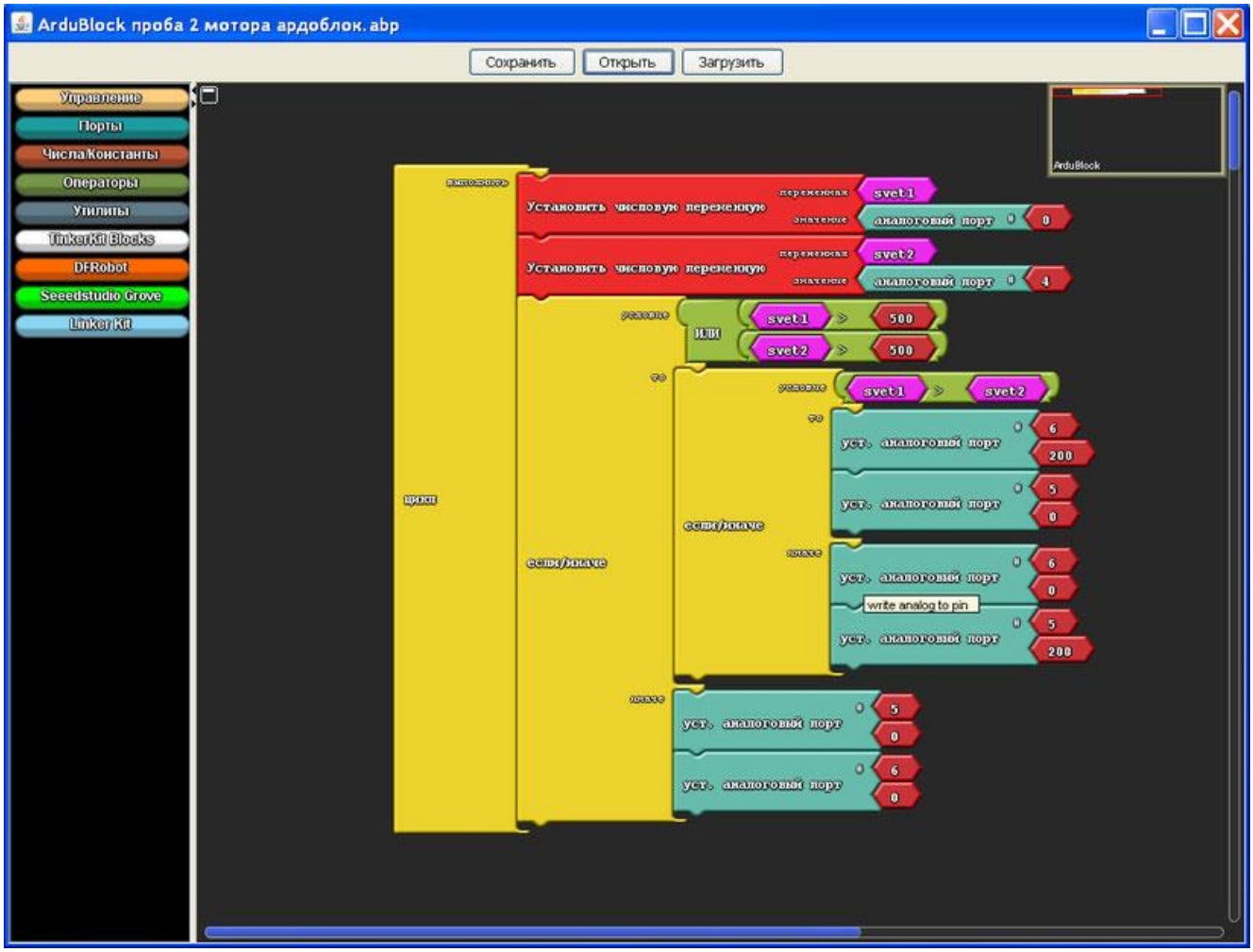


Рис. 2.1. Приклад створення програми (скетчу) для управління двома моторами на Ardublock.

Усі ці переваги призвели до того, що поступово відношення до платформи Arduino змінилося і переважна більшість людей, пов’язаних з електронікою (навіть професіоналів високого рівня), більше не вважає її чисто навчальною системою. Тепер усі зручності і переваги цієї системи використовують і професіонали, що розробляють реальні промислові рішення.

Таким чином, беручи до уваги усю вищенаведену інформацію, можна із впевненістю сказати, що платформа Arduino є найкращим варіантом у якості базового апаратного рішення для зведення системи моніторингу параметрів мікроклімату виробничих приміщень, що проектується.

За час розвитку платформи (тобто за 15 років з 2005 року, коли з'явився перший прототип плати Arduino) вийшло багато її версій, що зайняли свої ніші6 Arduino UNO, Nano, Mega, Leonardo і т.п. Однак, на сьогоднішній день найбільш універсальним рішенням є застосування версії Arduino UNO, яка і стає найбільш часто базою для реальних програмно-апаратних рішень – рис. 2.2.

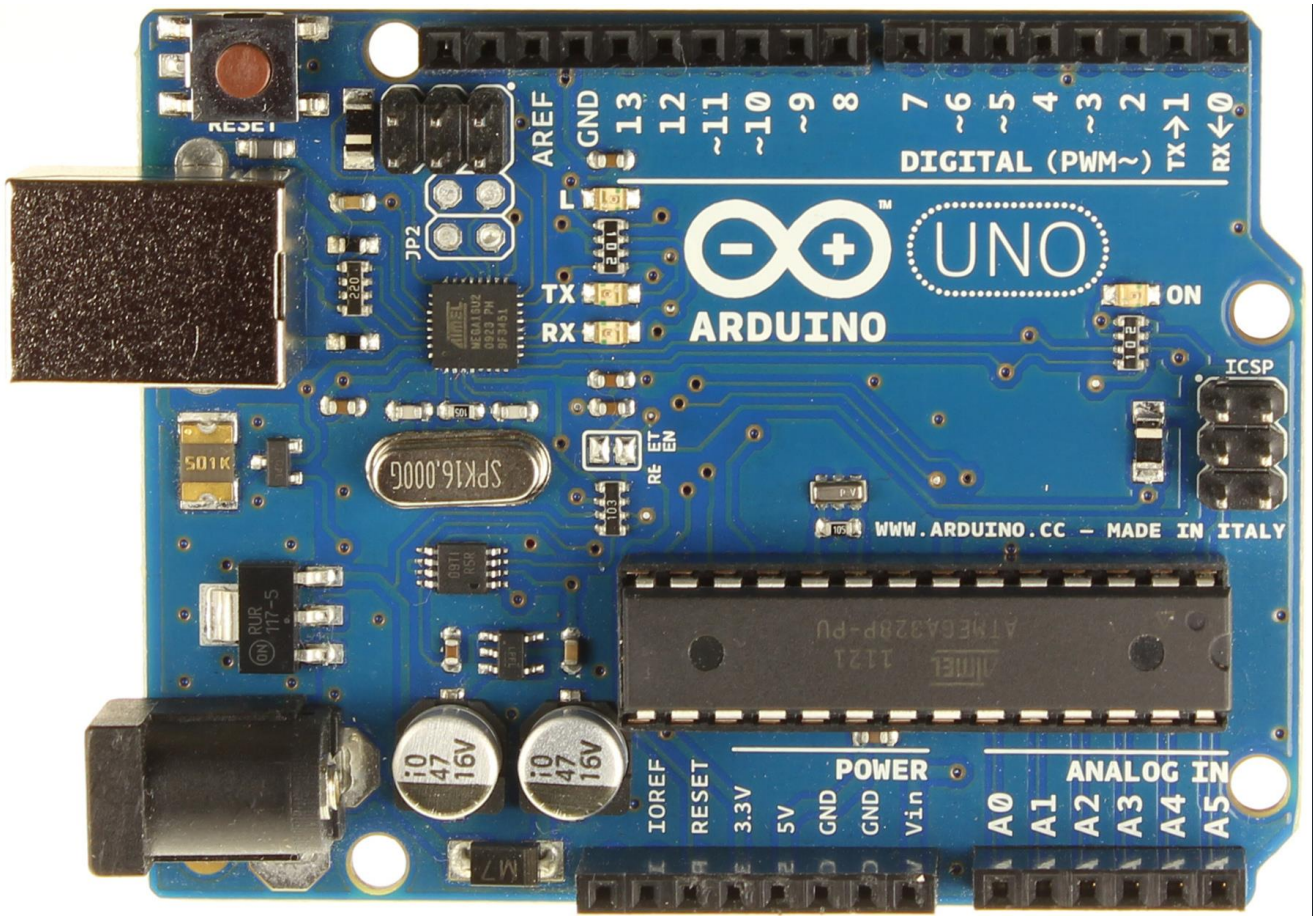


Рис. 2.2. Зовнішній вигляд найбільш популярної плати сімейства Arduino – UNO (оригінал, ревізія R3).

У нижній правій половині плати розміщено безпосередньо сам мікроконтролер ATmega328 у DIP корпусі (перевагою такої компоновки є можливість простої заміни мікроконтролера при його виході з ладу, що іноді трапляється при необережному використанні плати). З верхнього та нижнього країв розміщено цифрові та аналогові (ШІМ) порти, а також контакти живлення та управління по живленню. У лівій частині рисунку містяться роз'єми для підключення плати до шини USB (вгорі) та підведення стабілізованого живлення на 9 В (внизу). Зліва-вгорі кнопка перезавантаження мікроконтролеру. В цілому вся компоновка плати є дуже зручною та підготовленою до використання численних плат розширення (шилдів, від англ. shield), датчиків, виконуючих пристроїв і т.п., яких за роки активного розвитку платформи було випущено сотні (а може і тисячі).

Таким чином, у якості бази для зведення пристрою контролю параметрів мікроклімату приймаємо поширене та функціональне рішення на базі Arduino UNO.

2.2. Вибір типу та схеми розміщення датчиків системи моніторингу мікроклімату

Невід'ємною частиною будь-якого проекту Arduino, як і взагалі, будь-якого реального електронного пристрою, є підсистема датчиків, що знімають необхідні для роботи системи вхідні сигнали. Як було розглянуто в 1.2, необхідними датчиками є датчик температури та вологості. Таких елементів для Arduino було розроблено дуже багато, причому деякі з них – комбіновані (якраз для двох цих параметрів, оскільки вони досить часто вимірюються у зв'язці):

- DHT11, який вимірює температуру від 0°C до +50°C із точністю до 2% та вологість (тобто це комбінований датчик) в діапазоні 20-80% із точністю до 5%;

- DHT22, який вимірює температуру від –40°C до +125°C із точністю $\Delta T = \pm 0{,}5°C$ та вологість (це також комбінований датчик) в діапазоні 0-100% із точністю до 2%;

- DS18B20, який буває реалізований в одному із трьох різних форм-факторів (TO-92 - DS18B20, SO - DS18B20Z, μSOP - DS18B20U) і є дуже точним цифровим датчиком температури із діапазоном від –55°C до +125°C з точністю вимірювання до половини градусу, тобто $\Delta T = \pm 0{,}5°C$;

- TMP35, який вимірює температуру від 10°C до +125°C із точністю $\Delta T = \pm 2°C$, та його близький аналог TMP37;

- LM35, який вимірює температуру від –55°C до +150°C із точністю $\Delta T = \pm 0{,}5°C$, та його близький аналог LM335;

- AM2301, який вимірює температуру від –40°C до +80°C із точністю $\Delta T = \pm 0{,}5°C$ та вологість (тобто це комбінований датчик) в діапазоні 0-100% із точністю до 2%;

- GY-21 mini (Si7021), який вимірює температуру від –40°C до +125°C із точністю $\Delta T = \pm 0{,}4°C$ та вологість (тобто це комбінований датчик) в діапазоні 0-100% із точністю до 3%;

- тощо.

За умови використання датчика лише температури (не комбінованого), до нього слід приєднати датчик вологості повітря, серед яких можна назвати:

- датчик HR202 (резистивний), який вимірює вологість в діапазоні 20-90% із точністю до 2%;

- датчик HS1101 (ємнісний), який вимірює вологість в діапазоні 1-99% із точністю до 2%;

- тощо.

Аналізуючи ціни на вказану продукцію та її технічні характеристики, приходимо до висновку, що найкраще до умов задачі, що розв’язується, підходить комбінований датчик DHT22 – рис. 2.3, який при невисокій ціні має цілком задовільні для обраної галузі застосування характеристики (зокрема, відповідає умовам (1.2) та (1.3)).

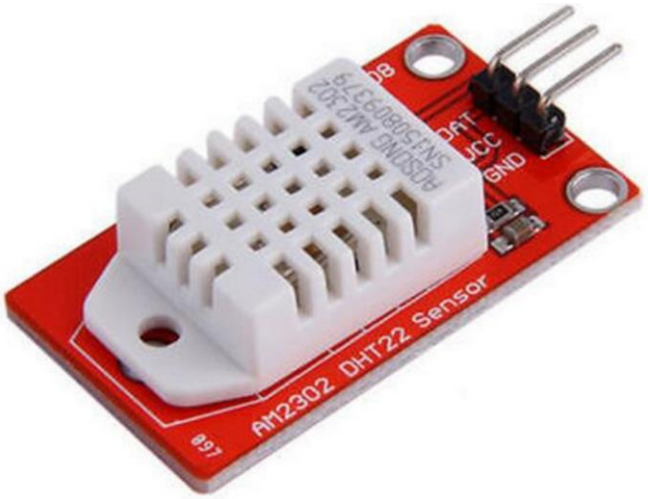
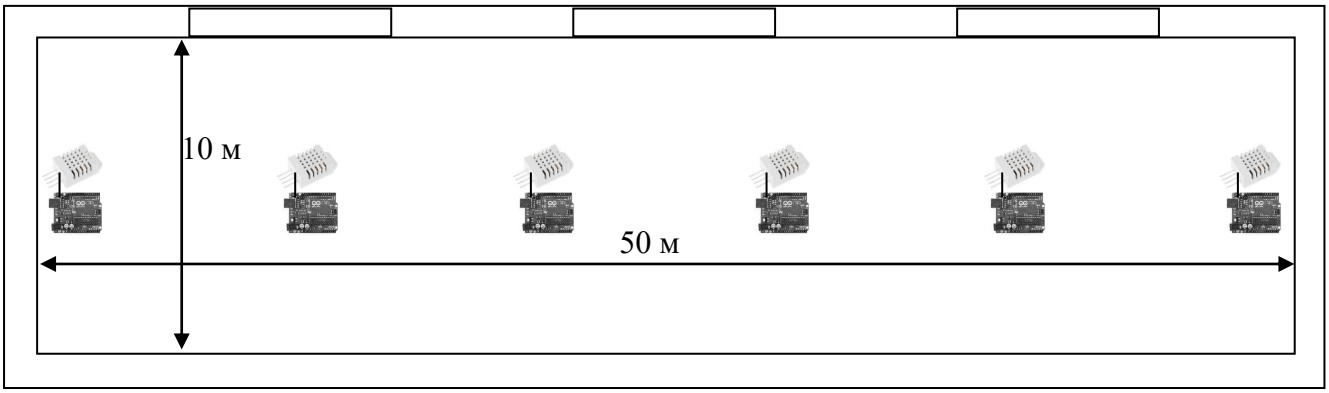
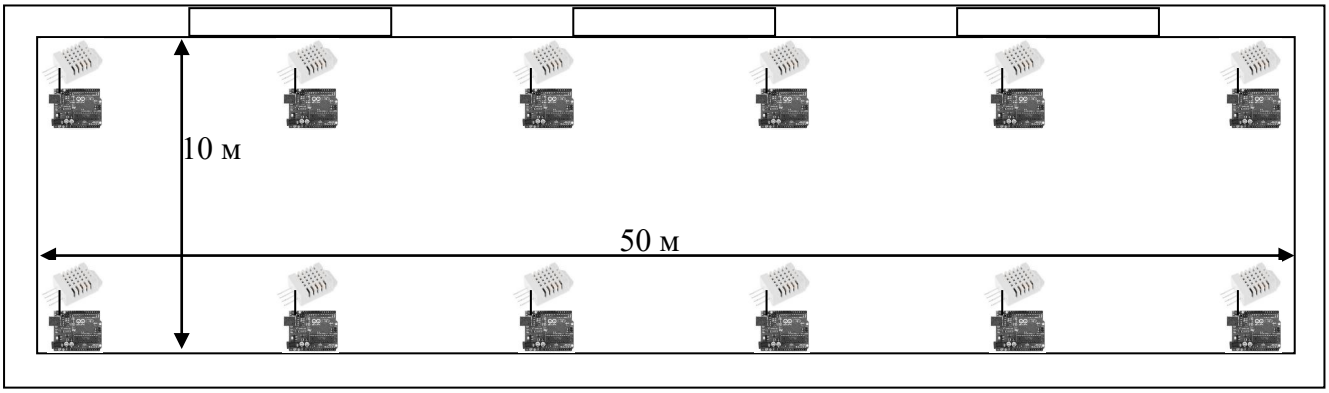


Рис. 2.3. Зовнішній вигляд комбінованого датчику DHT22 для вимірювання температури та відносної вологості.

Після вибору конкретної моделі датчику, постає питання про особливості його розміщення у кімнаті, де здійснюється моніторинг мікроклімату. Як уже зазначено вище, розміщувати один датчик у приміщенні з великою площею неефективно, тому слід обрати відстань, на якій доцільно розміщувати датчики (реалізовані в окремих, повністю незалежних пристроях, як було встановлено у розділі 1.2). Дуже мале значення відстані, порядку метра, неефективно, оскільки що у випадку охолодження, що нагрівання, при нормальній температурі приміщення у певній точці, такою ж буде і температура на відстані 1 м, задовільною буде і температура на відстані кілька метрів. Достатньою лінійною відстанню між двома сусідніми датчиками пропонується прийняти величину порядку 10 м, що в результаті розміщення їх у приміщенні розміром 10 м х 50 м (один із стандартних розмірів промислової теплиці, а також і великого холодильника) дає наступні варіанти схеми розміщення датчиків – рис. 2.4, *а* та *б*. Відмітимо, що через особливості будівельних норм та саму суть задачі забезпечення будівельної міцності, малоймовірним є використання приміщень із меншим лінійним розміром, що більший за 10 м (тобто 10 м вважатимемо максимальною величиною для меншого розміру будь-якого приміщення прямокутної форми: 10 м х 20 м, 10 м х 50 м, 10 м х 60 м і т.д. - допускаються, але не 15 м х 20 м, 20 м х 30 м, і т.п.).



а)



б)

Рис. 2.4. Дві пропоновані схеми розміщення датчиків: *а* – по середній лінії вздовж довгої сторони; *б* – по периметру приміщення (в обох випадках відстань між кожними двома сусідніми датчиками складає біля 10 м).

Зважаючи на подвоєння кількості датчиків при орієнтовно тій же ефективності, схему рис. 2.4, *б* використовувати не будемо (хіба що для якихось особливих, дуже критичних до змін температури застосувань), а у якості базового обираємо варіант розміщення 2.4, *а*. При такому способі розміщення датчиків два з них встановлюються біля середин двох протилежних стін приміщення, що є меншими за довжиною, у порівнянні з двома іншими протилежними стінами. Інші датчики розміщуються так, щоби відстань між двома сусідніми не перевищувала би 10 м. Зважаючи на те, що не всі лінійні розміри приміщень націло діляться на 10, для визначення кількості датчиків застосовуємо наступну формулу:

$$N = \left[\frac{a}{10} \right] + 2, \qquad (2.1)$$

де a – довжина приміщення (більший із двох лінійних його розмірів), м;

$[x]$ – позначення операції взяття цілої частини від аргументу x (тобто відкидання дробової частини).

Наприклад, якщо довжина приміщення складає 25 м, то за (2.1) кількість датчиків буде рівною 4, а відстань між двома сусідніми буде приблизно рівною 8,3 м. Якщо ж кількість датчиків була би рівною 3, то відстань між ними була би 12,5 м, що не задовольняє прийняту раніше вимогу, щоби максимальна відстань між двома датчиками дорівнювала би не більше 10 м.

Зафіксувавши тип та схему розміщення датчиків, можна переходити до визначення способу взаємодії їх із іншими частинами проектованої системи, інших частин системи між собою, а також і методу інформування користувача – одним словом до комунікаційних властивостей системи, що виконується у наступному підрозділі.

2.3. Комунікаційні властивості проектованої системи.

Під комунікацією в широкому сенсі зазвичай розуміють різні способи взаємодії елементів якоїсь системи. В даному випадку чітко виділяються наступні компоненти, між якими слід встановити комунікацію:

- датчик температури та вологості (комбінований, обраний вище DHT22);
- плата Arduino UNO;
- користувач системи, а саме його певний персональний технічний пристрій (наприклад, мобільний телефон, чи персональний комп’ютер, планшет, і т.д.) чи канал комунікації (наприклад, електронна пошта, SMS, Інтернет-месенджер, і т.д.).

Вибір способу зв’язку датчику DHT22 з платою Arduino є найбільш простим питанням, оскільки взагалі кажучи, він є строго фіксованим і здійснюється

шляхом простої електричної комутації виходів датчику із відповідними входами плати. На рис. 2.5 показано контакти (піни) датчику:

1 – контакт живлення на 5 В (VCC);

2 – контакт зчитування даних (DATA);

3 – не використовується (NC – Not Connected);

4 – контакт спільного проводу (заземлення, «земля», ground або GND).

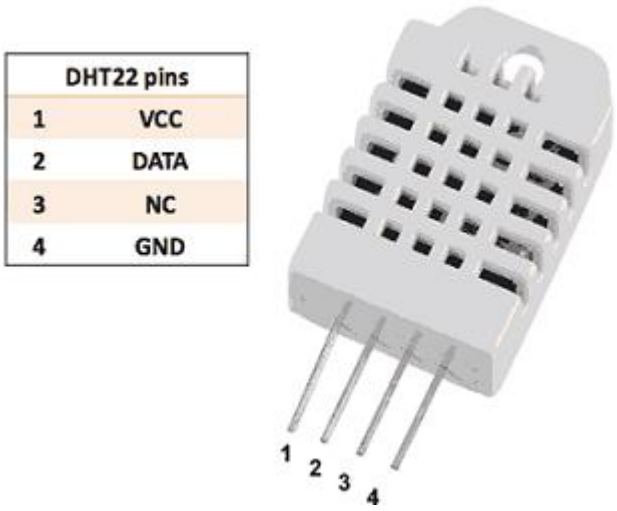
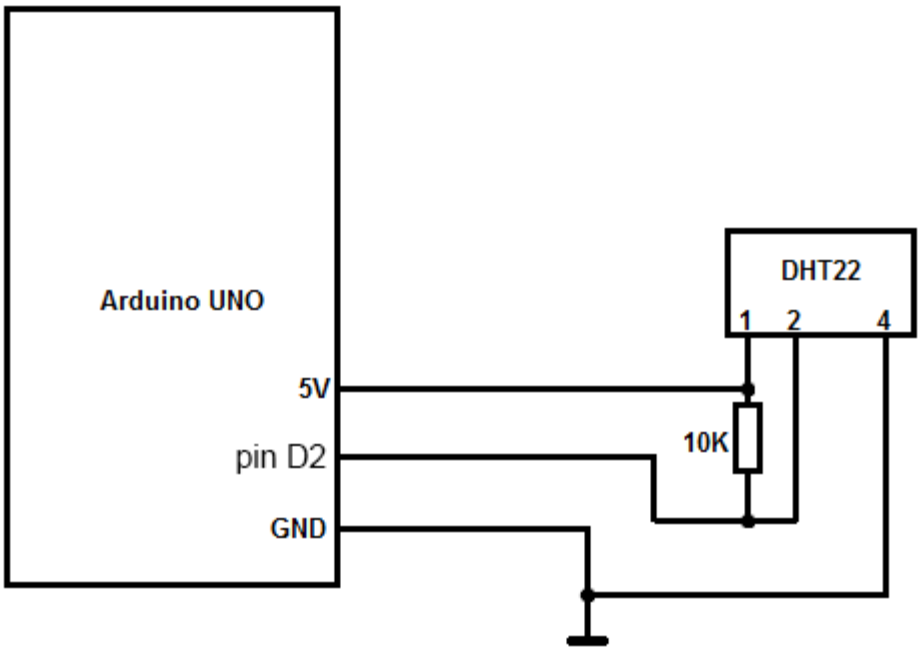
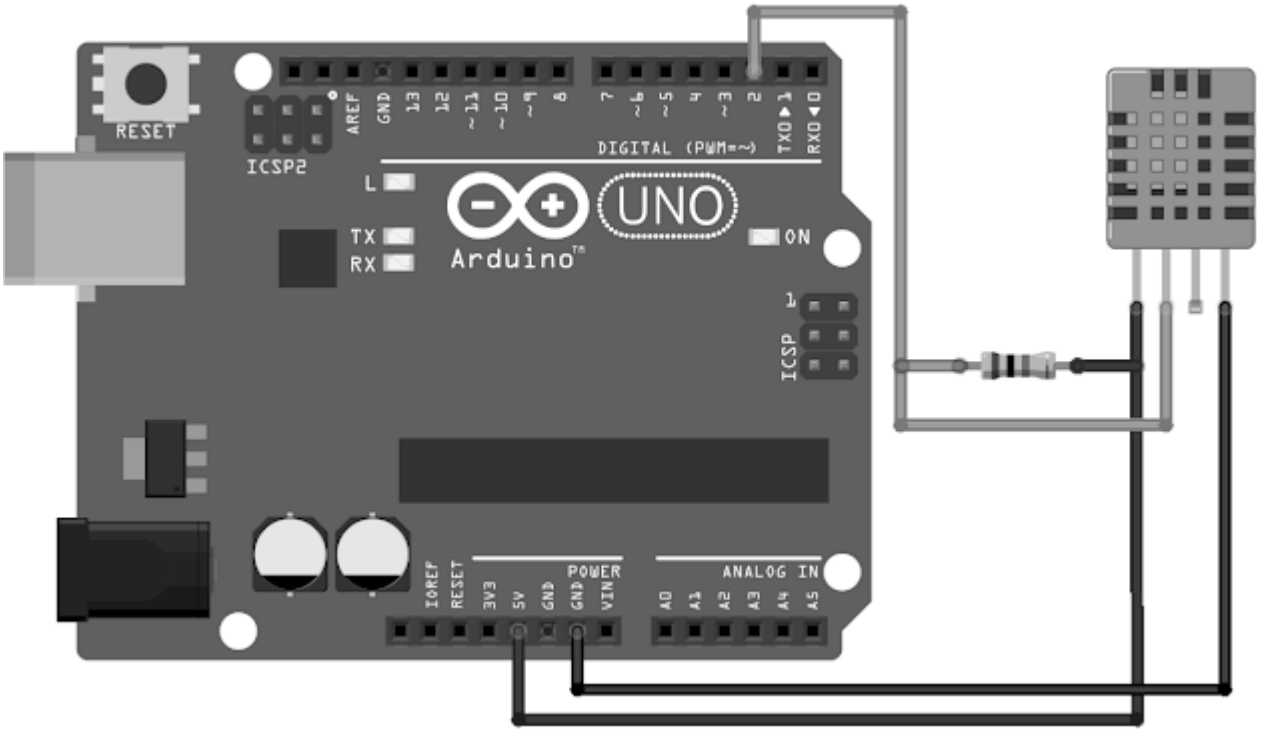


Рис. 2.5. Зовнішній вигляд датчику DHT22 та розшифровка його виходів.

Місце підключення контактів живлення 1 та 4 на платі Arduino є очевидним (відповідні контакти мають ці ж самі позначення), а вибору підлягає лише місце підключення контакту даних. Пін 2 можна підключити до будь-якого цифрового входу, наприклад D2. Для покращення якості роботи датчика, підвищення імовірності відсутності хибних значень у схему доцільно включити підтягуючий резистор, наприклад, на 10 кОм. Кінцева схема підключення датчика до плати Arduino матиме вигляд, як на рис. 2.6, *а* – принципова, 2.6, *б* - монтажна.



а)



б)

Рис. 2.6. Схеми підключення комбінованого датчику температури та вологості DHT22 до плати Arduino: *а* – принципова; *б* – монтажна.

На цьому комутацію вхідних елементів системи можна вважати завершеною, адже із розглянутого датчика контролер отримує усю необхідну для його роботи вхідну інформацію.

Наступним кроком є проектування комутації вихідних елементів системи, її виконуючих пристроїв. Оскільки проектування здійснюється для системи моніторингу (не контролю або управління), то основною її метою є інформування людини, користувача цієї системи (насправді мова іде лише про передачу інформації на певний персональний пристрій, що є у доступності відповідальної за це людини, тому у перспективі вихідний каскад, що здійснює інформування, може бути підключений до якогось автоматизованого пристрою, що здійснює корегування параметрів роботи системи і без участі людини, принаймні якісь первинні, основні такі дії). Інформування людини можна виконувати надсилаючи інформацію на наступні електронні цифрові пристрої:

- персональний комп’ютер (ноутбук);
- планшет;
- смартфон;
- мобільний телефон у загальному сенсі (без мультимедійних можливостей та встановлених ОС).

Надіслати інформацію на ПК (ноутбук) можна засобами електронної пошти, чи якоїсь програми-месенджера (на зразок Viber, WhatsApp, Telegram, ICQ чи QIP), однак людина не є прив’язаною до власного ПК чи ноутбуку, і може, як мінімум, на цілий день іти займатися іншими справами, пропускаючи важливе повідомлення від системи моніторингу. Відповідно, ПК чи ноутбук не є пристроями, на які можна надсилати критичну інформацію, що потребує термінової реакції.

Планшет на сьогоднішній день не є пристроєм, яким забезпечені буквально усі шари населення, тим більше із розвитком можливостей сучасних смартфонів кількість користувачів такого пристрою, як планшет, поступово спадає, тому його розглядати не будемо.

Смартфон є зручним засобом, на який досить легко (дешево з фінансової точки зору та просто з технічної) відправити повідомлення, причому як текстове, так і мультимедійне, у вигляді картинки, звукозапису чи відео. До того ж існує чимало різних каналів комунікації, за допомогою яких можна зробити таку відправку. Крім того, смартфон майже постійно знаходиться у фокусі сучасної людини, лише на невеликі проміжки часу, та й то дуже рідко, виходячи із зони її видимості. Однак, суттєвим недоліком смартфону є порівняно низька поширеність цього засобу. Сучасна молодь майже на 100% забезпечена більш, чи менш продуктивними смартфонами, але люди середнього та старшого віку використовують їх поки що недостатньо. В той же час великою є імовірність того, що кінцевим споживачем інформації від системи моніторингу параметрів мікроклімату (теплиці чи холодильнику) буде не осучаснена просунута молода людина, а сторож-пенсіонер, або інша особа похилого віку, яка можливо мешкає поруч із приміщенням і може взяти на себе обов'язки у неробочий час, або святкові дні якнайшвидше дістатися об'єкту у разі потреби та розпочати процес стабілізації цих контрольованих показників у разі тривожного повідомлення від системи моніторингу. Відповідно, смартфон у якості кінцевого пристрою системи розглядати також, нажаль, не можна.

Останнім пристроєм із переліку наведеного вище, є звичайний мобільний телефон (смартфон, або ні), будь-яка модель якого має функцію прийому голосових дзвінків та текстових повідомлень SMS. Поширеність мобільних телефонів на сьогоднішній день серед соціалізованого населення є близькою до 100% незалежно від віку, тому даний пристрій найбільш доцільно використовувати у якості цілі для пересилання тривожних інформаційних повідомлень від системи моніторингу параметрів мікроклімату до відповідальної особи. Технічно більш зручно та надійно організувати інформування у вигляді SMS, для чого потрібно впровадити у проектовану систему GSM-модуль зв'язку.

Для платформи Arduino доступними є наступні варіанти GSM-модулів:

- GPRS/GSM+GPS шилд на SIM808, вартістю біля 30 у.о.;
- GSM / GPRS SIM800 модем вартістю біля 14 у.о.;

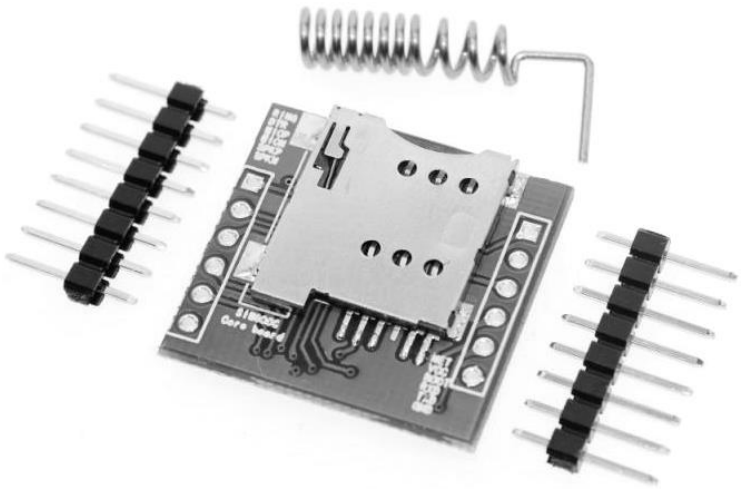
- SIM900 GSM/GPRS модуль вартістю біля 30 у.о.;

- SIM800C V2 модуль GSM 850/900/1800/1900 вартістю біля 7 у.о.;

- GSM-модуль SIM800C з антеною вартістю біля 7 у.о., тощо.

Останній із приведених варіантів є повноцінним пристроєм, що дозволяє пересилати текстові повідомлення, тобто відповідаючи потребам даного проекту, має найнижчу ціну, а додатковою перевагою має можливість роботи у мережах всіх існуючих діапазонів частот (850, 900, 1800 та 1900 мегагерц). Відмітимо, що, на відміну від передостаннього варіанту, у вказану ціну включається вартість антени – рис. 2.7, *б*. Відповідно, у даній роботі будемо використовувати модуль SIM800C, зовнішній вигляд якого наведено на рис. 2.7.

а)



б)

Рис. 2.7. Зовнішній вигляд обраного GSM-модуля SIM800C: *а* – фронт; *б* – зад.

Як видно на рис. 2.7, *а* на модулі присутній роз'єм для SIM-карти формату microSIM, тому для кожного пристрою слід використовувати власну SIM-карту відповідного розміру.

Важливою особливістю усіх GSM модулів є специфічні вимоги до величини сигналів у вольтах. Так, наприклад, живлення має складати від 3,4 до 4,4 В, що є не зовсім стандартним діапазоном і може забезпечуватися як шляхом впровадження дільника напруги, або іншого стандартного способу зниження її значення. Те ж саме стосується і логічної одиниці, тобто високого рівня, величина якого складає 2,8 В, що в принципі також може забезпечуватися згаданим способом. Однак, більш доцільним для отримання нестандартних значень напруги є використання спеціального модуля, що, окрім пониження її значення, може ще здійснювати і стабілізацію на заданому рівні. Наприклад, такими модулями є:

- понижуючий конвертер на мікросхемі MP1584 з максимальним струмом 3А, вхідною напругою 4,5–28В, вихідною 0,8–20В (задається резистором підлаштування на платі);

- понижуючий конвертер на базі LM2596S з вхідною напругою 4,5–40В, вихідною 0,8–20В (задається резистором підлаштування на платі);

- тощо.

Оскільки вибір модуля-конвертера взагалі не є важливим етапом процесу проектування (оскільки можна взагалі обійтися і без нього, використавши всього лише дільник напруги), то не приділяючи цьому процесу багато уваги, оберемо перший-ліпший модуль MP1584. У такому випадку схема підключення обраного GSM-модуля буде мати вид рис. 2.8.

Мінімально необхідно для вирішення поставленої задачі задіяти чотири наступних піни:

- живлення VCC;

- земля GND;

- отримання даних RXD;

- пересилки даних TXD.

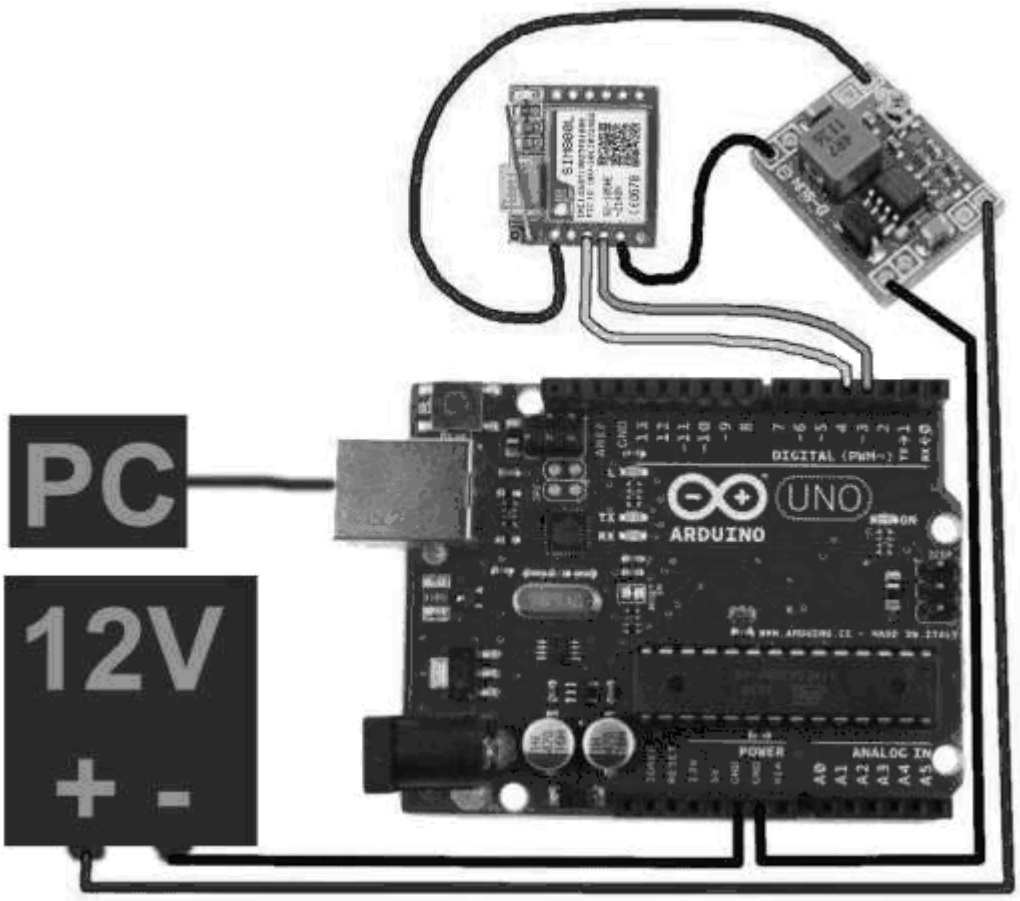


Рис. 2.8. Монтажна схема підключення GSM-модуля SIM800C через стабілізатор MP1584.

Контакти живлення та заземлення підключаємо до відповідних виходів на платі Arduino, а вхід та вихід GSM-модуля для передачі даних з’єднуємо з портами 3 та 4 (оскільки порт 2 вже зайнятий під датчик температури та вологості DHT22 відповідно до схеми рис. 2.6).

Вказаних з’єднань на рис. 2.6 та рис. 2.8 достатньо для зведення готового пристрою та його програмування, що і буде виконуватися у наступному розділі.

2.4 Висновки по розділу

Таким чином, у даному розділі здійснено обґрунтування вибору апаратної платформи, що була обрана у якості бази для зведення системи моніторингу параметрів мікроклімату виробничих приміщень. А саме, рішення будемо

виконувати на базі платформи Arduino, яка за час свого розвитку поступово перейшла із області навчальних продуктів до цілком реальних промислових. У розділі шляхом порівняльного аналізу обрано датчики системи та модуль зв’язку а також схеми їх підключення. Запропоновано схему розміщення датчиків у приміщенні та обґрунтовано рішення відмовитися від централізованої системи, реалізовуючи усю систему моніторингу у вигляді набору цілком автономних, самодостатніх пристроїв для вимірювання температури і вологості та здійснення тривоги при їх виході за межі допустимого діапазону. Керуючись прийнятими особливостями апаратної частини системи, можна переходити до створення її програмної складової, що виконується у наступному розділі.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ МОНІТОРИНГУ МІКРОКЛІМАТУ ВИРОБНИЧИХ ПРИМІЩЕНЬ

3.1. Вибір типу та особливостей інтерфейсу користувача проектованої системи

Для будь-якої технічної системи реалізація взаємодії із людиною є важливим елементом всього процесу проектування. Для програмної системи розробка інтерфейсу грає надзвичайно важливу роль і оформлена цілим розділом галузі IT таким, як UI (User Interface – інтерфейс користувача). При проектуванні технічних (апаратних) пристроїв важливу роль відіграють їх ергономічні характеристики, що проявляються у наявності певних регуляторів, перемикачів, кнопок, рубильників – для управління станом пристрою, та у присутності екранів, індикаторів, лампочок і т.п. – для відображення його поточного стану. Для системи, що проектується, також потрібно реалізувати інтерфейс, який очевидно, завжди ділиться на дві частини:

- засоби введення інформації від користувача в систему для управління особливостями її роботи;

- засоби виведення інформації користувачеві про параметри поточного стану системи.

В деяких випадках один і той же елемент управління може використовуватися як для введення даних, так і для виведення, але в даному випадку такої специфіки не спостерігається.

Отже, введення інформації в систему відбувається від датчиків (датчику), але не від користувача. Фактично єдина вхідна інформація (налаштування), що потребує система – це відомості про дозволений діапазон температури та відносної вологості повітря у контрольованому приміщенні. Ці величини задаються один раз і далі не змінюються, оскільки великі виробничі приміщення, оснащені недешевими системами охолодження чи (специфічного) нагріву, рідко змінюють власне призначення і у такому, екстраординарному, випадку, що

потребує значних затрат та численних операцій, перепрограмування системи моніторингу параметрів мікроклімату не представляє суттєвого труда. Таким чином, дозволені діапазони вхідних параметрів, що підлягають моніторингу, пропишемо безпосередньо у висхідний код програмного забезпечення системи, а впроваджувати якісь елементи управління для налаштування цих параметрів під час роботи системи не будемо (для уникнення необґрунтованого підвищення вартості всієї системи). При необхідності зміни цих діапазонів будемо перепрограмовувати усі пристрої системи, що не займе більше 5-10 хвилин на кожен пристрій.

Таким чином, строго кажучи елементи управління для введення інформації від користувача в систему у проектованому рішенні відсутні.

Вихідні параметри системи моніторингу – це значення поточної температури та вологості, зняті по різним точкам контрольованого приміщення. Ці значення можна було би, звичайно, контролювати безпосередньо, підходячи до кожного датчика та передивляючись значення із екрану пристрою, однак такий підхід повністю зводить нанівець усі переваги дистанційного моніторингу з елементами інтелектуальної обробки даних, адже потребує постійної присутності людини на об’єкті (при цьому взагалі ніякі системи моніторингу не потрібні, а достатньо забезпечити працівника звичайним термометром). Як уже було розібрано вище, даний підхід не є ефективним, як з організаційної, так і з фінансової точки зору. Отже, вихідні параметри слід передавати дистанційно, причому у попередньому розділі було запропоновано використовувати для цього повідомлення SMS.

Очевидно, що надсилати ці повідомлення не слід постійно, оскільки пам’ять будь-якого телефону швидко переповниться (особливо простих моделей, на які власне і розрахована дана система) і наступні повідомлення просто не будуть завантажуватися пристроєм відповідальної особи. Для підвищення ефективності інформування відповідальної особи, система моніторингу повинна відповідати наступним вимогам:

- по-перше, надсилати повідомлення в обов'язковому порядку потрібно лише у випадку тривоги, тобто виходу контрольованих параметрів за допустимі межі;

- по-друге, навіть при тривожній ситуації, слід ввести певний (не дуже малий) тайм-аут між двома послідовними повідомленнями, що обумовлено тими ж самими міркуваннями незахаращення телефону відповідальної особи повідомленнями (взагалі, в інструкцію користувача системи слід ввести правило видалення найстаріших повідомлень від системи моніторингу, коли загальна кількість SMS на телефоні наближається до максимального значення) – обираємо інтервал відправлення тривожних повідомлень рівним 5 хвилин;

- по-третє, текст тривожного повідомлення сформувати по схемі:

«Температура на датчику 3 рівна 23 град. при нормі від 6 до 12»,

або

«Вологість на датчику 1 рівна 90% при нормі від 60 до 75»;

- по-четверте, для впевненості, що система працює нормально (реально здійснює моніторинг і фіксує дозволені значення температури та вологості, а не вийшла з ладу і просто не може нічого відправити), слід іноді (дуже рідко, пропонується інтервал у 8 годин, що рівний одній зміні) відправляти повідомлення статусного характеру із текстом:

«За останні 8 годин температура та вологість на датчику 2 у межах норми»;

- у випадку, якщо відповідальна особа не вчиняє потрібних запобіжних дій (або вони не ефективні) і параметри системи не повертаються у допустимі діапазони протягом тривалого інтервалу часу (зокрема, пропонується величина цього інтервалу, рівна одній годині, оскільки за цей час можна встигнути зробити певні дії, направлені на виправлення шкідливої ситуації, а якщо незадовільні параметри не змінюються, то продукція може розпочинати псуватися і треба інформувати вище керівництво), то системі моніторингу доцільно відправити тривожне повідомлення ще і директору компанії із текстом:

«Більше години на датчику 3 фіксуються порушення мікроклімату!».

Відмітимо, що для підвищення універсальності системи вже неодноразово приймалися інженерні рішення, що забезпечували б прийом повідомлень відповідальними співробітниками за допомогою мобільних телефонів самих старих моделей. Але такі пристрої можуть не підтримувати кирилицю (тобто відсилку повідомлень українською мовою). Крім того, кількість символів при відправці кирилицею є вдвічі меншою, ніж при відправці латиницею, тому розглянуті повідомлення не вміщуються у цю стандартну довжину при використанні української абетки. Якщо ж надсилати їх «транслітом», то кожне з розглянутих вище повідомлень, вміщується у межі одного SMS, що є додатковим плюсом такого рішення. Отже, вказані у цьому підрозділі повідомлення будемо надсилати транслітом.

Відповідно, інтерфейс системи полягає у відправці вказаних повідомлень на телефони відповідальної особи (перша дія, що повторюється кожні 5 хвилин за час безперервного порушення допустимих параметрів мікроклімату), та директора підприємства (друга, однократна дія).

3.2. Вибір інструментальних засобів для розробки

Наступним кроком при розробці системи моніторингу є вибір (та його обґрунтування – за наявності кількох альтернатив) засобів, за допомогою яких буде здійснюватися розробка програмного забезпечення системи моніторингу, що створюється. Першим, найбільш загальним питанням при цьому стає вибір технології програмування, оскільки саме вона визначає наступні кроки по вибору більш конкретних засобів, зокрема і допоміжних.

3.2.1. Обґрунтування вибору технології програмування

Першочерговим питанням, яке постає перед розробниками практично будь-якого програмного забезпечення, є вибір моделі або технології його розробки. Такими, що реально використовуються на сьогоднішні день у виробничій

практиці, є технології структурного (процедурного) та об’єктно-орієнтованого програмування. Кожна з них має свої особливості, переваги і недоліки, які розглянемо докладніше.

Структурне, або як його ще називають практикуючі програмісти, процедурне програмування засноване на використанні окремих структурних блоків - в першу чергу, підпрограм (процедур і функцій).

Історично перші комп’ютерні програми були відносно простими і мали пакетний режим роботи: отримуючи на вхід якусь інформацію (можливо, навіть на перфокарті) вони виконували певний обсяг операцій з обробки цих даних і видавали результат. При цьому існувала сувора функціональна залежність виходу від входу – рис. 3.1.

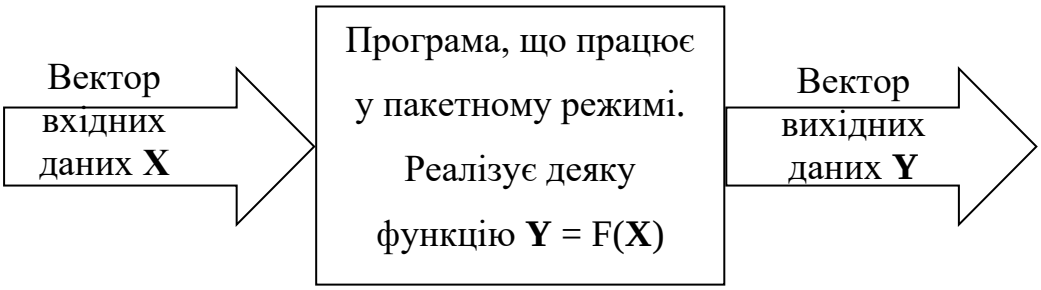


Рис. 3.1. Схема роботи пакетної програми.

Незважаючи на відсутність явного зв’язку функціональності і структури, пакетні програми в основному мали просту лінійну послідовність виконання. Тобто в них практично відсутні будь-які підпрограми, принаймні, використання підпрограм не було наріжним каменем самої методики програмування.

В міру ускладнення функціональності програмного забезпечення, змінювалася і його внутрішня структура: поступово розвинулася інтерактивна модель взаємодії користувача і програми – рис. 3.2. Програми стали запитувати інформацію і активно реагувати на дії людини. Ускладнення функціональності привело до відповідного ускладнення програмного коду, якого, в першу чергу, стало просто багато. Багато для того, щоб людина-програміст без всяких спеціальних хитрувань швидко і легко розібралася з незнайомим кодом.

Розміщувати код у простій лінійній послідовності без виділення великих блоків стало незручно, в першу чергу, для розуміння цього коду.

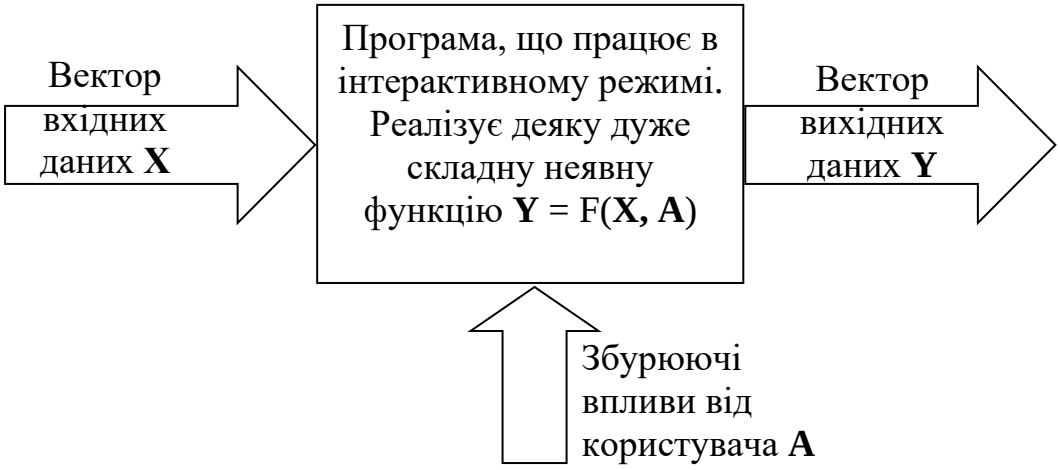


Рис. 3.2. Схема роботи програми, що активно взаємодіє з користувачем.

Тут слід зазначити психологічні особливості сприйняття людиною складних «великих» завдань. Неструктуроване «велике» завдання (наприклад, написання дипломної роботи) зазвичай викликає певний психологічний ступор і, як результат, повну неможливість поступово розібратися з ним. Людині зручно розбити проблему на не надто велику (зазвичай до десятка, а краще 3-4) кількість завдань (наприклад, розділів у дипломній роботі), не замислюючись про реалізацію кожного з них. Коли є ясність і розуміння проблеми на найвищому рівні абстракції, слід приступати до деталізації підзадач, кожну з яких слід розбити на окремі «підпідзадачі» тобто підзадачі нижчого рівня, більш дрібні. Уже після такого розбиття слід аналізувати всі перераховані підзадачі. На певному етапі зупиняються і виконують не розбиття чергової підзадачі на більш дрібні, а безпосередню її реалізацію в програмних кодах.

Отже, можна сказати, що розвиток методики програмування відбувався в ногу з розвитком призначеного для користувача інтерфейсу: і грубо кажучи, розвиненому інтерфейсу командного рядка відповідає парадигма структурного програмування.

Говорячи більш строго, структурне програмування має на увазі побудова програми відповідно до трьох основних принципів: слідування, розгалуження, повторення.

Слідування має на увазі, що оператори і блоки програмного коду слідують і виконуються один за іншим. Розгалуження реалізується різними умовними операторами типу if, і дозволяє вибирати один з декількох подальших варіантів виконання програми. Повторення зазвичай відносять на рахунок циклів (що йдуть підряд багаторазових повторів одного і того ж ділянки коду), хоча цей же принцип можна віднести і до підпрограм.

Взагалі ж, структурне програмування у деякій мірі є застарілою методикою програмування, на зміну якій разом з віконним інтерфейсом прийшло об'єктно-орієнтоване програмування (деякі сучасні мови програмування загального призначення навіть не дозволяють створити структурну програму, тільки об'єктно-орієнтовану – як, наприклад, Visual C# чи Java). Проте, при створенні невеликих програм (наприклад, до 10000 рядків коду і без передбачуваного розширення) застосування цієї методики програмування більш виправдано і код краще сприймається, ніж його об'єктно-орієнтований варіант.

Суть же методології об'єктно-орієнтованого програмування полягає в тому, що система розглядається, як сукупність окремих сутностей - об'єктів, які мають набір якихось своїх внутрішніх параметрів - властивостей, а також можуть взаємодіяти між собою за допомогою деяких дій - викликів методів (або трохи більше непрямим чином - шляхом надсилання повідомлень, оброблюваних методами об'єктів; для цього необхідна присутність активної сутності, яка роздає повідомлення адресатам, як, наприклад, менеджер вікон в ОС Windows).

Якщо говорити про програмний код, то для того, щоб оперувати об'єктом, його спочатку потрібно створити. Об'єкти створюються як змінні, у яких типом виступає клас об'єкта. Клас - це просто опис, які властивості можуть мати об'єкти такого типу (тобто яку інформацію вони можуть зберігати), і які у них є методи (тобто які дії вони можуть виконувати). Об'єкт - це набір значень, чому саме рівні

властивості даного об’єкта (свої методи кожен об’єкт отримує від свого класу, тобто методи однакові у всіх об’єктів, що належать даному класу).

Для чого потрібен цей специфічний підхід, адже самі по собі об’єкти не додають нічого корисного (навпаки, введення об’єктів ускладнює програму, вносить в неї нові сутності)? Виявляється, реалізовуючи всі сутності, необхідні, згідно з алгоритмом, для роботи програми, у вигляді класів і об’єктів, ми спрощуємо її розуміння для самих себе. Саме тому ОО-підхід рекомендується до застосування для великих проектів (більше десятків тисяч рядків коду), коли утримувати «в голові» всю систему цілком стає важко. Можна сказати, що розбиття програми на об’єкти і проектування їх класів наближає розуміння предметної області до звичного людського образу мислення (в разі «великих» проектів). Людина мислить класами, об’єктами і зв’язками між ними.

Порівняльна складність проектів, що мають однакову функціональність, але побудованих по-різному (згідно структурному і об’єктно-орієнтованого підходів до програмування), як функція їх обсягу показана на рис. 3.3. З графіка рис. 3.3 слід, що, якщо потрібно реалізувати продукт з невеликою функціональністю (тобто кількість рядків коду, що її реалізують буде очевидно невеликою, порядку кількох тисяч рядків), то це краще робити без застосування класів, так як вони будуть тільки ускладнювати всю справу. Якщо ж програма має більш-менш значну функціональність, а значить, реалізується хоча б кількома тисячами рядків коду, то вже є сенс замислюватися про застосування об’єктно-орієнтованого підходу. Однозначно будуватися за принципами ООП повинні програми, які мають 10000 рядків коду і більш.

Відзначимо, що часто крім розглянутих міркувань, також на вибір методики програмування впливають інші чинники, наприклад, можливість майбутнього розширення функціональності, створення якомога більш зрозумілого коду (для роботи над проектом цілої команди, а не одного програміста), або просто побажання замовника застосувати найбільш сучасний підхід до програмування.

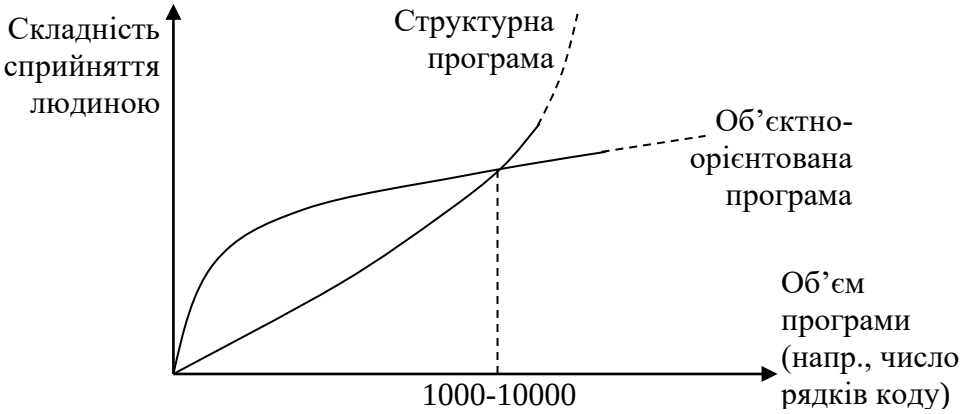


Рис. 3.3. Порівняльна складність висхідного тексту двох програм, що мають однакову функціональність, але реалізованих по-різному: згідно об'єктно-орієнтованому та структурному підходам.

Крім розбиття (декомпозиції) всієї предметної області на об'єкти (класи) і співвідношення між ними, також ОО-підхід має на увазі дотримання трьох основних його принципів: інкапсуляція, наслідування, поліморфізм.

Під інкапсуляцією мається на увазі об'єднання даних (значення властивостей класу у деякого конкретного об'єкта) та засобів їх обробки (методи класу). Це знову ж таки зручно психологічно, так як дозволяє реалізовувати окремі завершені сутності - класи, які самі обробляють свої дані. Звернення до об'єктів цих класів відбувається за допомогою методів, що утворюють інтерфейс класу.

Наслідування дуже корисно, тому що дозволяє сильно скоротити обсяги повторюваного коду (до чого потрібно завжди прагнути при розробці будь-якого програмного забезпечення). Згідно з цим принципом виділяється клас, який має загальний набір властивостей і методів для декількох більш розширених класів. Цей клас оголошується батьком, базовим класом для декількох похідних від нього (нащадків, спадкоємців). Всі класи-нащадки успадковують від базового всі його властивості та методи, але до цього ще мають свої власні оригінальні властивості і / або методи.

Наприклад, клас Студент є похідним від класу Людина, тому що кожна людина має властивість Ім’я, Прізвище, метод Відпочити(). Однак у Студента є свої специфічні властивості і / або методи, які як раз і відрізняють його від просто Людини: СереднійБал, НомерЗаліковки, ЗдатиЕкзамен(), і т.д.

При наслідуванні іноді методи батьківського і похідного класу мають однакове призначення, але реалізовуються по-різному. Такі методи називаються перевантаженими. Наприклад, метод Відпочити() у класу Людина реалізується як відпочинок на дивані, а у класу Студент - як похід в клуб. При цьому ще раз підкреслимо, що призначення методу в обох випадках одне і той саме.

Поліморфізм є можливістю деякої функції приймати об’єкти як батьківського, так і похідних класів, і вміти викликати перевантажені методи саме того класу, об’єкт якого був переданий в функцію. Слід сказати, що це досить специфічна можливість і в загальному багато програмістів використовують ОО-підхід і без звернення до поліморфізму.

Нехай, наприклад, у програмі є функція ПровестиВихідні(), припустимо яка не належить якомусь класу (хоча це не принципово). Нехай аргументом цієї функції є об’єкт класу Людина. Тоді в неї можна передавати об’єкти всіх похідних від Людини класів: Студент, Службовець, Пенсіонер, і т.д., тому що всі вони є Людиною (спадкоємці цього класу). Ясно, що ця функція повинна включати різні дії: ПрибратиКвартиру(), ПітиНаРинок(), і в тому числі Відпочити(). Так ось поліморфізм дозволяє всередині цієї функції просто вказати назву методу Відпочити(), не вказуючи якого саме класу він повинен бути викликаний, а вже в процесі виконання програми, якщо в функцію переданий об’єкт класу Студент, то викликається саме його метод Відпочити(), а якщо переданий об’єкт класу Пенсіонер, то автоматично викликається саме його метод Відпочити(), і т.д. Кажуть, що функція ПровестиВихідні() - поліморфна, і вона є такою завдяки тому, що реалізує принцип поліморфізму.

Важливими поняттями в ООП також є: статичні члени класу, абстрактні методи і класи, дружба функцій і класів, і т.д.

Грунтуючись на перерахованих особливостях двох існуючих методів програмування, вибираємо структурний підхід як більш простий, що відповідає невеликим (не промисловим) масштабам проектованого ПЗ, а також особливостям предметної галузі. Також (і це головне) структурний підхід прекрасно реалізується засобами розробки для платформи Arduino, а саме у середовищі розробки Arduino IDE.

3.2.2. Вибір мови програмування

Для обраної технології розробки можна використати мови програмування C/C++, Python, Go, Basic, графічні мови сімейства Scratch (Ardublock).

Одразу виключимо використання графічних засобів розробки, оскільки вони мають однозначний імідж навчальних засобів для молодіжної аудиторії. Проекти для реального промислового використання не можуть створюватися за допомогою графічних засобів. Також не будемо розглядати мову програмування Basic, яка у всі часи мала статус навчальної (як і Паскаль), а на сьогоднішній день у великій мірі вважається застарілою.

Мова програмування Go (Golang) має чимало переваг у порівнянні з іншими існуючими мовами програмування. Вона розроблялася як мова програмування для створення високоефективних програм, що працюють на сучасних розподілених системах і багатоядерних процесорах. Вона може розглядатися як спроба створити заміну мов C/C++ з урахуванням змінених комп'ютерних технологій і накопиченого досвіду розробки великих систем. Go був розроблений для вирішення реальних проблем, що виникають при розробці програмного забезпечення в Google. У якості основних таких проблем можна назвати:

- повільну збірку програм;
- неконтрольовані залежності;
- використання різними програмістами різних підмножин мови;
- труднощі з розумінням програм, викликані незручностями у читанні коду, поганим документуванням і так далі;

- дублювання розробок;

- високу вартість оновлень;

- несинхронні поновлення при дублюванні коду;

- складність розробки інструментарію;

- проблеми міжмовної взаємодії.

Основними вимогами до мови стали наступні.

а) Ортогональность. Мова повинна надавати невелике число засобів, які не повторюють функціональність один одного.

б) Проста і регулярна граматика. Мінімум ключових слів, проста, легко розбирається граматична структура, легко читається код.

в) Проста робота з типами. Типізація повинна забезпечувати безпеку, але не перетворюватися на бюрократію, лише збільшує код. Відмова від ієрархії типів, але зі збереженням об'єктно-орієнтованих можливостей.

г) Відсутність неявних перетворень.

д) Прибирання сміття.

е) Засоби розпаралелювання, прості та ефективні.

є) Підтримка рядків, асоціативних масивів і комунікаційних каналів.

ж) Чіткий поділ інтерфейсу і реалізації.

з) Ефективна система пакетів з явною вказівкою залежностей, що забезпечує швидку збірку.

Go створювався в розрахунку на те, що програми на ньому будуть транслюватися в об'єктний код і виконуватися безпосередньо, не вимагаючи віртуальної машини, тому одним з критеріїв вибору архітектурних рішень була можливість забезпечити швидку компіляцію в ефективний об'єктний код і відсутність надмірних вимог до динамічної підтримки.

В результаті вийшла мова, «яка не стала проривом, але тим не менш стала відмінним інструментом для розробки великих програмних проектів».

Хоча для Go доступний і інтерпретатор, практично в ньому немає великої потреби, так як швидкість компіляції досить висока для забезпечення інтерактивної розробки.

Розглянемо основні можливості мови Go.

а) Go - мова з суворою статичної типізацією. Доступно автоматичне виведення типів, для типів користувача - «качина типізація».

б) Повноцінна підтримка покажчиків, але без можливості застосовувати до них арифметичні операції, на відміну від C/C++/D.

в) Строковий тип з вбудованою підтримкою Unicode.

г) Використання динамічних масивів, хеш-таблиць (словників), зрізів (слайсів), варіант циклу для обходу колекції.

д) Засоби функціонального програмування: неіменовані функції, замикання, передача функцій в параметрах і повернення функціональних значень.

е) Автоматичне управління пам'яттю зі складальником сміття.

є) Засоби об'єктно-орієнтованого програмування, але без підтримки наслідування реалізації (успадковуються тільки інтерфейси). За великим рахунком, Go є процедурною мовою з підтримкою інтерфейсів.

ж) Засоби паралельного програмування: вбудовані в мову потоки (go routines), взаємодія потоків через канали та інші засоби організації багатопоточних програм.

з) Досить лаконічний і простий синтаксис, заснований на C, але істотно доопрацьований, з великою кількістю синтаксичного цукру.

Go не містить цілого ряду популярних синтаксичних засобів, доступних в інших сучасних мовах прикладного програмування, що викликано свідомим рішенням розробників задля виключення потенційно небезпечних (неоднозначних) можливостей платформи.

Головним же недоліком мови Go є її мала поширеність, а, отже, і наявність слабкої підтримки серед спільноти розробників, тим більше для такої платформи, як Arduino. Ще одна доступна для Arduino мова програмування – Python.

Python (читають як «пайтон» або «пітон») – відносно молода мова програмування загального призначення. Це означає, що писати цією мовою можна усе, що завгодно, у т.ч. і програми для мікроконтролерів. Розробниками Python зазначається, що саме ядро мови є надзвичайно компактним (цим

забезпечується висока швидкість програм, написаних на Python), але його супроводжують стандартні бібліотеки з широкими функціональними можливостями (а це дозволяє використовувати дану мову для будь-яких цілей та вхідних даних).

Слід відмітити, що Python дозволяє створювати програми з використанням різних парадигм програмування: структурного, об’єктно-орієнтованого, функціонального, імперативного та аспектно-орієнтованого. Звичайно, у більшості проектів використовується структурний (для порівняно малих програм) та об’єктно-орієнтований (ефективний для програм, що містять десятки тисяч рядків коду та більше) підхід до програмування.

Синтаксис мови Python в цілому С-подібний, але у деяких моментах спрощений (наближається до мінімалістичного). Безсумнівною перевагою є те, що інтерпретатор Python реалізований практично на всіх платформах і операційних системах. Першою такою мовою був С, проте його типи даних на різних машинах могли займати різну кількість пам'яті і це служило деякою перешкодою при написанні дійсно переносимої програми. Python же таким недоліком не володіє.

Наступна важлива риса - розширюваність мови, цьому надається велике значення і, як пише сам автор, мова була задумана саме як розширювана. Це означає, що є можливість вдосконалення мови всіма зацікавленими програмістами. Інтерпретатор написаний на С і вихідний код доступний для будь-яких маніпуляцій. У разі необхідності, можна вставити його в свою програму і використовувати як вбудовану оболонку. Або ж, написавши на С свої доповнення до Python і скомпілювавши програму, отримати "розширений" інтерпретатор з новими можливостями.

Наступна перевага - наявність великого числа модулів, що підключаються до програми і забезпечують різні додаткові можливості. Такі модулі пишуться на С і на самому Python і можуть бути розроблені усіма досить кваліфікованими програмістами.

Єдиним недоліком мови є порівняно невисока швидкість виконання Python-програми, що обумовлено її інтерпретованістю. Однак, на наш погляд, це з

лишком окупається перевагами мови при написанні програм не дуже критичних до швидкості виконання.

Розглянемо особливості мови Python.

1. Python, на відміну від багатьох мов (Pascal, C++, Java, і т.д.), не вимагає опису змінних. Вони створюються в місці їх ініціалізації, тобто при першому присвоєнні змінної будь-якого значення. Значить, тип змінної визначається типом значення, що присвоюється. В цьому відношенні Python нагадує Basic.

Тип змінної не є незмінним. Будь-яке присвоювання для неї коректно і це призводить лише до того, що типом змінної стає тип нового значення, що присвоюється.

2. У таких мовах як Pascal, C, C++ організація списків представляла деякі труднощі. Для їх реалізації доводилося добре вивчати принципи роботи з покажчиками і динамічною пам'яттю. І навіть маючи гарну кваліфікацію, програміст, кожен раз заново реалізуючи механізми створення, роботи і знищення списків, міг легко допустити важковловимі помилки. Зважаючи на це, були створені деякі засоби для роботи зі списками. Наприклад, в Delphi Pascal є клас TList, який реалізує списки; для C++ розроблена бібліотека STL (Standard Template Library), що містить такі структури як вектори, списки, множини, словники, стеки і черги. Однак, такі засоби є не у всіх мовах і їх реалізаціях.

Однією з відмінних рис Python є наявність таких вбудованих в саму мову структур як тьюпли (tuple), списки (list) і словники (dictionary), які іноді називають картами (map).

Python на відміну від Pascal, C, C++ не підтримує роботу з покажчиками, динамічною пам'яттю і адресну арифметику. У цьому він схожий на Java. Як відомо, покажчики служать джерелом важковловимих помилок і робота з ними відноситься більше до програмування на низькому рівні. Для забезпечення більшої надійності і простоти вони не були включені в Python.

Однією із особливостей Python є те, як відбувається присвоювання однієї змінної іншій, тобто коли по обидва боки від оператора "=" стоять змінні.

Будемо називати семантикою покажчиків випадок, коли присвоювання призводить лише до присвоюванню посилання (покажчика), тобто нова змінна стає лише іншим ім'ям, що позначає ту саму ділянку пам'яті, що і стара змінна. При цьому зміна значення, що позначається нової змінної, призведе до зміни значення старої, тому що вони, фактично, означають одне і те ж.

Коли ж присвоювання призводить до створення нового об'єкта (тут об'єкт - в сенсі ділянки пам'яті для зберігання значення будь-якого типу) і копіювання в нього вмісту привласнюється змінною, цей випадок назвемо семантикою копіювання. Таким чином, якщо при копіюванні діє семантика копіювання, то змінні по обидві сторони від знака "=" означатимуть два незалежних об'єкта з однаковим змістом. І тут наступна зміна однієї змінної ніяк не позначиться на іншій.

Присвоєння в Python відбувається наступним чином: якщо об'єкт, що присвоюється, є екземпляром таких типів як числа або рядки, то діє семантика копіювання, якщо ж в правій частині стоїть екземпляр класу, список, словник чи тьюпл, то діє семантика покажчиків.

Недоліком мови Python, який перешкоджає повноцінному її використанню для проектів на базі Arduino, також є мала кількість інформаційних ресурсів, присвячених цьому питанню.

Основним засобом за допомогою якого уже багато років пишуть усі професійні програми світу, в тому числі і системне ПЗ для мікроконтролерів, в т.ч. і для платформи Arduino, є C/C++. У випадку з платформою Arduino і не об’єктно-орієнтованою технологією розробки буде використовуватися мова С.

3.2.3. Вибір середовища розробки та додаткових програмних засобів

Оскільки розробці підлягає програмний продукт (скетч) для конкретної архітектурної платформи (Arduino), то вибір засобів розробки є дещо обмеженим. Можна використати наступні середовища розробки:

- Arduino IDE;

- Programino;

- B4R (Basic for Arduino);

- CodeBlocks for Arduino.

Класичним засобом, що пропагується розробниками платформи від самого початку її використання є Arduino IDE, тому його і будемо використовувати для потреб даного дослідження. Саме для цього випадку (Arduino IDE + мова C) в наявності присутня величезна кількість інформаційних матеріалів на будь-які потрібні випадки. Наявність розвинутого ком'юніті на сьогоднішній день є важливим доводом при виборі технологій, мов та засобів розробки (і взагалі – будь-яких інструментів).

Відмітимо, у якості додаткової інформації, що середовище Programino є платним, B4R використовує на сьогоднішній день досить рідкісну мову програмування Basic, а середовище CodeBlocks підтримує значно більше можливостей, ніж тільки програмування для Arduino, тому має високий поріг входження (є складним у використанні, підходить тільки для тривалого використання на постійній основі). Ці фактори додатково показують доцільність вибору саме обраного IDE.

В цілому, середовище Arduino IDE є надзвичайно зручним, має порівняно простий інтерфейс (особливо для тих осіб, хто має досвід використання сучасних середовищ типу Visual Studio чи IntelliJ IDEA) та досить повні допоміжні інструменти, завдяки чому і набуло великої популярності серед розробників платформи Arduino.

3.3. Формалізація алгоритмічної складової програмно-апаратного комплексу, що розробляється

Алгоритми роботи пристроїв та систем унаочнюються графічними блок-схемами, а також можуть подаватися у текстовій описовій формі. Зафіксуємо алгоритм роботи одного вимірювально-інформувального пристрою системи:

1) після включення виконати блок ініціалізації мікроконтролеру, а саме:

а) проініціалізувати датчик температури;

б) проініціалізувати послідовний порт для видачі діагностичних повідомлень у тому випадку, якщо пристрій будуть тимчасово підключати до ПК для виконання діагностики;

в) проініціалізувати GSM-модуль.

2) У традиційному для платформи Arduino циклі loop виконувати наступні дії:

а) Скинути прапорець тривоги wasalarm;

б) Зчитати із датчика поточні значення температури та вологості;

в) Якщо не вдалося успішно зчитати ці значення, тобто датчик вийшов з ладу, відправити повідомлення про необхідність його заміни відповідальній особі та відключитися на одну зміну (8 годин);

г) Видати отримані значення температури та вологості у серійний порт на ПК;

д) Якщо температура виходить за дозволені межі, то відправити SMS-повідомлення відповідальній особі та встановити прапорець тривоги wasalarm;

е) Якщо вологість виходить за дозволені межі, то відправити SMS-повідомлення відповідальній особі та встановити прапорець тривоги wasalarm;

є) Якщо прапорець тривоги скинутий, тобто обидва параметри під час останнього вимірювання опинилися у допустимих межах, то обнулити лічильник підряд зафіксованих тривожних спрацьовувань numconsalar і збільшити на одиницю лічильник підряд зафіксованих нормальних вимірювань numnormmeas;

ж) Інакше (тобто якщо прапорець тривоги встановлений), тобто якщо хоча б один із параметрів під час останнього вимірювання вийшов за допустимі межі, то збільшити на одиницю лічильник підряд зафіксованих тривожних спрацьовувань numconsalar і обнулити лічильник підряд зафіксованих нормальних вимірювань numnormmeas;

з) Якщо зафіксовано більше 12 підряд тривожних спрацьовувань системи, то обнулити лічильник numconsalar і відправити тривожне повідомлення

директору компанії, з інформацією про те, що вже годину на об’єкті існує проблема з мікрокліматом і її не можуть виправити;

и) Якщо зафіксовано більше 96 підряд нормальних вимірювань параметрів мікроклімату, то обнулити лічильник numnormmeas і відправити статусне повідомлення відповідальній особі, що протягом останніх 8 годин на об’єкті усе гаразд;

і) Зачекати 5 хвилин до наступного вимірювання.

Оскільки алгоритм роботи не має складних розгалужень чи циклів, то для виконання програмної реалізації цілком достатньо наведеного текстового опису (за вимогою замовника системи, йому може бути надана блок-схема цього алгоритму), і можна переходити до імплементації проекту у програмних кодах.

3.4. Особливості програмної реалізації системи моніторингу мікроклімату виробничих приміщень

В першу чергу, надамо перелік змінних, які використовуються у проекті:

- об’єкт mySerial типу SoftwareSerial – для роботи з двома пінами, до яких підключені провідники, які ідуть до GSM модуля, як зі звичайним серійним портом (тобто два цих піни призначаємо як RX та TX);

- цілочисельна змінна delaytime для збереження затримки у 5 хвилин = = 300 секунд = 300 000 міллісекунд;

- об’єкт dht типу DHT – для роботи з датчиком температури;

- цілочисельні змінні lowT, highT – для збереження нижньої та верхньої допустимих температур;

- цілочисельні змінні lowH, highH – для збереження нижнього та верхнього допустимих значень вологості повітря;

- цілочисельний номер датчику sID, ця змінна має бути різною для усіх окремих пристроїв, що утворюють систему моніторингу. Даний номер слід нанести на корпус відповідного пристрою, а також відобразити на схемі розміщення датчиків типу рис. 2.4;

- цілочисельна змінна numnormmeas - кількість підряд зафіксованих нормальних вимірювань (тобто і температура і вологість під час такого вимірювання у межах норми);

- цілочисельна змінна numconsalar - кількість підряд зафіксованих порушень мікроклімату (тобто або температура, або вологість, або обидва ці параметри під час такого вимірювання вийшли за межі норми);

- цілочисельна змінна wasalarm – прапорець, який сигналізує, чи було порушення необхідних умов мікроклімату під час останнього вимірювання: 1 – було, 0 – не було;

- локальна для функції loop() змінна humidity – зберігає зчитане з датчика значення вологості повітря;

- локальна для функції loop() змінна temperature – зберігає зчитане з датчика значення температури;

- допоміжна змінна i – лічильник циклу.

Розглянемо також деякі особливості програмного коду, що реалізує алгоритм із підрозділу 3.3. Так, у програмі широко використовується процедура відправки SMS, яка реалізується кодом на зразок:

```
    Serial.println("SMS everything OK started");
    mySerial.println("AT+CMGS=\"" + (String)"+380999999999" +"\"");
    delay(500);
    mySerial.print("Za ostanni 8 godyn temperatura ta vologist na
sensori"+(String)sID+" u mezhah normi");
    delay(500);
    mySerial.print((char)26);
    delay(500);
    Serial.println("SMS everything OK complete");
    delay(2000);
```

Спочатку тут у звичайний серійний порт відправляється повідомлення про старт відправлення СМС та його призначення, наприклад, у наведеному тут конкретному випадку відправляється повідомлення про те, що система протягом

останніх 8 годин працює у нормальному режимі. Ця інформація потрібна у тому випадку, якщо пристрій підключений до ПК і здійснюється його діагностика.

Сам процес відправки повідомлення виконується за допомогою посилання на GSM-модуль так званої AT-команди (ці команди використовуються для спілкування та управління роботою телефонних пристроїв: модемів, модулів, цифрових телефонів; AT-команди також використовуються у функції setup() для вибору текстового режиму – CMGF - та кодування - CSCS - при пересиланні SMS). Код команди для відправлення SMS – CMGS, після чого вказується номер телефону отримувача, текст повідомлення та завершальні символи.

Після відправлення кожної команди на GSM-модуль задається певна відстрочка, оскільки виконання команди на пристрої вимагає часу, а у фіналі на діагностичний серійний порт відправляється текстовий рядок, що SMS успішно надіслано.

В цілому, текст програми не є складним та добре читається. Повний текст розміщено у Додатку А.

3.5. Документаційне забезпечення розробленого програмно-апаратного комплексу

3.5.1. Інструкція про розгортання апаратної частини системи

По-перше, на основі геометричних параметрів приміщення, у якому має контролюватися мікроклімат, за формулою (2.1) визначається необхідна кількість окремих пристроїв N, що формують систему моніторингу, яка розробляється. Визначившись із кількістю датчиків N, слід зібрати таку кількість окремих пристроїв відповідно до схем рис. 2.6 та 2.8, забезпечивши їх автономними елементами живлення на 9 В (батарейка типу «Крона», що традиційно використовується для живлення платформи Arduino). Далі слід запрограмувати пристрої відповідно до Додатку А. При цьому кожному пристрою слід призначити власний номер 1, 2, …, N. Відповідно, при компіляції скетча слід для

кожного датчика змінювати значення змінної sID. Номер пристрою записати на його поверхні. Отримані пристрої, відповідно до схеми рис. 2.4, *а*, слід розмістити у виробничому приміщенні. Перевірити відповідність розміщення датчиків (за їх номерами) строго по схемі рис. 2.4, де мають бути підписані їх номери. Подати живлення на усі пристрої (вставити елементи живлення, або увімкнути їх перемикачем, якщо при паянні була реалізована така можливість). Система готова до використання.

3.5.2. Інструкція адміністратора створеної системи

Існує декілька ситуацій, що потребують втручання адміністратора системи, а саме:

а) при отриманні повідомлення про неможливість зчитування температури та вологості необхідно:

- відповідно до схеми розміщення датчиків типу рис. 2.4 знайти датчик, що ніби-то зламався;
- перевірити фізичну цілісність пристрою, усіх видимих з’єднань та елементів;
- спробувати замінити елемент живлення на новий;
- у випадку, якщо позбутися проблеми власними силами не вдалося, зв’язатися з розробниками системи.

б) при отриманні хибного тривожного повідомлення користувачем (який отримав тривожне повідомлення, дістався об’єкта та виявив, що порушень мікроклімату насправді не було), він звертається до адміністратора, якому необхідно:

- відповідно до схеми розміщення датчиків типу рис. 2.4 знайти датчик, що подав хибний сигнал;
- перевірити фізичну цілісність пристрою, усіх видимих з’єднань та елементів;
- спробувати замінити елемент живлення на новий;

- підключити плату Arduino до ноутбуку та завантажити Arduino IDE, запустити програму моніторингу послідовного порту;

- спробувати подіяти на датчик фізичними впливами: піднести полум’я запальнички або сірника і встановити підвищення температури, піднести лід із морозильної камери холодильника та зачекати декілька хвилин, визначаючи її зниження, подмухати з близька на датчик, від чого має підвищитися вологість – якщо реакції системи немає – сповістити розробника про вихід датчика з ладу

- у випадку, якщо позбутися помилки власними силами не вдалося, зв’язатися з розробниками системи.

3.5.3. Інструкція користувача розробленої системи

Нарешті, найменш кваліфікованою людиною, що працює із системою є користувач, і він має дотримуватися наступної інструкції:

- тримати весь час своєї зміни мобільний телефон у включеному стані, причому у такому режимі, що дозволяє отримувати SMS-повідомлення, за що він несе персональну відповідальність;

- читати усі повідомлення, що приходять від датчиків системи;

- у випадку отримання статусного повідомлення про нормальну роботу системи (кожні 8 годин) слід видалити найбільш старе повідомлення від цього ж абонента для забезпечення наявності вільного місця для отримання майбутніх SMS;

- у випадку отримання тривожного повідомлення слід негайно вирушити безпосередньо на об’єкт передбаченим для цього способом, попередньо узгодженим із керівництвом підприємства (власним транспортом, пішки, бігом, на таксі, і т.п.);

- при потраплянні на об’єкт здійснити візуальне спостереження та оцінити спочатку власними органами чуттів, а потім - за допомогою термометра температуру на об’єкті, шукати можливі джерела порушення мікроклімату: розбиті вікна, відкриті двері, порушення стелі та стін; шукати непрацююче

кліматичне обладнання: перевірити живлення кожного пристрою, цілісність корпусів та провідників, здійснити повний візуальний контроль приміщення та обладнання;

- у випадку знаходження причини відхилень, спробувати усунути її власноруч і, якщо це вдалося, впевнитися у нормальній роботі системи, повідомити про ситуацію керівництво та залишити об’єкт;

- якщо причина є складною і усунути її власноруч неможливо, слід повідомити про це керівництво і терміново звернутися до спеціалістів (на підприємства, що здійснюють ремонтні роботи), дочекатися їх приїзду та подальших інструкцій керівництва;

- якщо візуальний і температурний контроль не виявив відхилень від нормальної роботи, а спрацьовування системи було викликано лише одним пристроєм, повідомити про це адміністратора системи, вказавши йому номер пристрою, що здійснив хибне спрацьовування, та залишити об’єкт.

3.6. Тестування та випробування розробленої системи моніторингу мікроклімату

Для тестування системи було зібрано один пристрій відповідно до схем рис. 2.6 та 2.8, та запрограмовано його відповідно до Додатку А. Численні випробування різними температурами (феном, льодом із морозильника, диханням та паром із ванної кімнати) показали адекватну роботу системи та відправлення відповідних SMS-повідомлень - рис. 3.4.

Послідовність випробувань була наступною:

- спочатку пристрій піднесли до працюючого фену і отримали перше повідомлення (о 12:11);

- потім із морозильника звичайного побутового холодильника назбирали льоду і помістили пристрій на лід, залишивши його на декілька хвилин, в результаті чого отримали наступне повідомлення (о 12:23);

- далі розмістили сенсор біля поверхні дуже гарячої води у великій ємності (каструлі), яку прикрили кришкою, в результаті чого отримали третє повідомлення (о 12:33);

- далі залишили пристрій у холодильній камері на 3 хвилини, в результаті чого отримали четверте повідомлення про нормальні параметри мікроклімату протягом заданого часу спостереження.

Рис. 3.4. Повідомлення, надіслані системою моніторингу, що підтверджують її адекватну роботу.

Слід відмітити, що для тестування адекватності роботи системи при нормальних показниках мікроклімату (четверте повідомлення) тимчасово був використаний скетч із зменшеним значенням для порівняння змінної numnormmeas (1 замість 96). При цьому 102 рядок програми мав вигляд:

if(numnormmeas>1)

Це дозволило не чекати 8 годин до отримання повідомлення про нормальну роботу системи, а витратити всього лише 5 хвилин при тих же умовах, тобто робота система у цьому випадку може вважатися вірною.

В цілому, система працює стабільно, без збоїв, адекватно виконуючи власну функцію – контроль параметрів мікроклімату.

3.7. Висновки по розділу

Таким чином, у даному розділі здійснено розробку програмного забезпечення системи моніторингу мікроклімату виробничих приміщень. В першу чергу, спроектовано інтерфейс системи, який здійснюється шляхом відправлення інформаційних та тривожних SMS-повідомлень. Обрано технологію розробки (структурне програмування), мову програмування (традиційну для програмування мікроконтролерів – C) та середовище розробки (Arduino IDE). Розроблено алгоритм роботи системи, що має досить простий лінійний вид із невеликими розгалуженнями, залежно від потрапляння контрольованих параметрів мікроклімату у задані діапазони. Розглянуто важливі аспекти програмної реалізації проекту, а саме докладно проаналізовано змінні, що застосовані у проекті, та особливості техніки виконання деяких стандартних операцій, що виконуються програмним забезпеченням (зокрема, особливості відправки SMS-повідомлень). Також у розділі створено мінімальний комплект документації, необхідний для експлуатації системи, а саме:

- інструкції по розгортанню системи;
- інструкцію адміністратора;
- інструкцію користувача.

Описується тестування системи, яке показало її адекватну роботу, а саме те, що при виході контрольованих параметрів за допустимі межі система реально надсилає тривожні повідомлення, причому діє у відповідності до розроблених протоколів та алгоритму дій.

ВИСНОВКИ

Таким чином, у даній роботі здійснено проєктування та реалізацію автоматизованої системи моніторингу параметрів мікроклімату виробничих приміщень, що являє собою програмно-апаратне рішення із програмою написаною мовою C/C++.

В першу чергу проаналізовано особливості видів виробничої діяльності, що потребують контролю мікроклімату та встановлено вимоги до автоматизованих систем, що можуть його здійснювати. За основу для такої системи обрано дуже популярну на сьогодні платформу Arduino, яка за час свого довгого розвитку із розряду навчальних систем перейшла до категорії промислових. Обрано необхідні датчики системи, яким став дешевий, але досить функціональний DHT22, та комунікаційне обладнання, яким обрано модуль SIM800C. Розроблена структура пристрою, що є базовою автономною одиницею із всього їх набору, що в цілому і утворює систему моніторингу. Було розроблено алгоритм роботи системи та реалізовано відповідне програмне забезпечення, необхідне для виконання системою своїх функцій. Система пройшла тестування, яке показало належне виконання нею задач, покладених на неї відповідно до мети дослідження. В цілому робота може вважатися виконаною, а мета дослідження досягнутою, оскільки розроблена система дійсно оперативно повідомляє відповідальних осіб про виявлені проблеми з мікрокліматом контрольованого виробничого приміщення.

У перспективі система може розвиватися шляхом додавання елементів управління пристроєм для динамічної зміни параметрів його роботи (замість того, щоби прошивати ці значення у програмний код системи), що однак несе її подорожчання.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Banzi M., Shiloh M. Getting Started with Arduino. 3rd ed. Sebastopol: Maker Media, 2014. 262 p.
2. Blum J. Exploring Arduino: Tools and Techniques for Engineering Wizardry. 2nd ed. Hoboken: Wiley, 2019. 384 p.
3. Monk S. Programming Arduino: Getting Started with Sketches. 2nd ed. New York: McGraw-Hill Education, 2016. 208 p.
4. Schmidt J. Arduino: A Technical Reference. Sebastopol: O’Reilly Media, 2016. 254 p.
5. Richardson M., Wallace S. Getting Started with Raspberry Pi and Arduino. Sebastopol: O’Reilly Media, 2012. 180 p.
6. Boxall J. Arduino Workshop: A Hands-On Introduction with 65 Projects. San Francisco: No Starch Press, 2013. 392 p.
7. McRoberts M. Beginning Arduino. 2nd ed. New York: Apress, 2013. 424 p.
8. Evans B., Noble J. Arduino in Action. Shelter Island: Manning Publications, 2013. 328 p.
9. Pecori R. Arduino: La guida ufficiale. Milano: Apogeo, 2012. 250 p.
10. Fedoruk G. Системи автоматизації на базі Arduino: навчальний посібник. Львів: Видавництво Львівської політехніки, 2020. 148 с.
11. Kozak O. Мікроконтролери Arduino в освітніх проектах: навчальний посібник. Київ: НТУУ «КПІ», 2019. 132 с.
12. Norris B. Arduino for Dummies. Hoboken: Wiley, 2013. 400 p.
13. Blum J. Exploring Arduino: Tools and Techniques for Engineering Wizardry. Hoboken: Wiley, 2013. 336 p.
14. Margolis M. Arduino Cookbook. 3rd ed. Sebastopol: O’Reilly Media, 2020. 796 p.
15. Karvinen T., Karvinen K., Valtokari V. Make: Arduino Bots and Gadgets. Sebastopol: O’Reilly Media, 2011. 296 p.

16. Di Justo P., Richardson M. Make: Arduino Projects. Sebastopol: O'Reilly Media, 2012. 174 p.

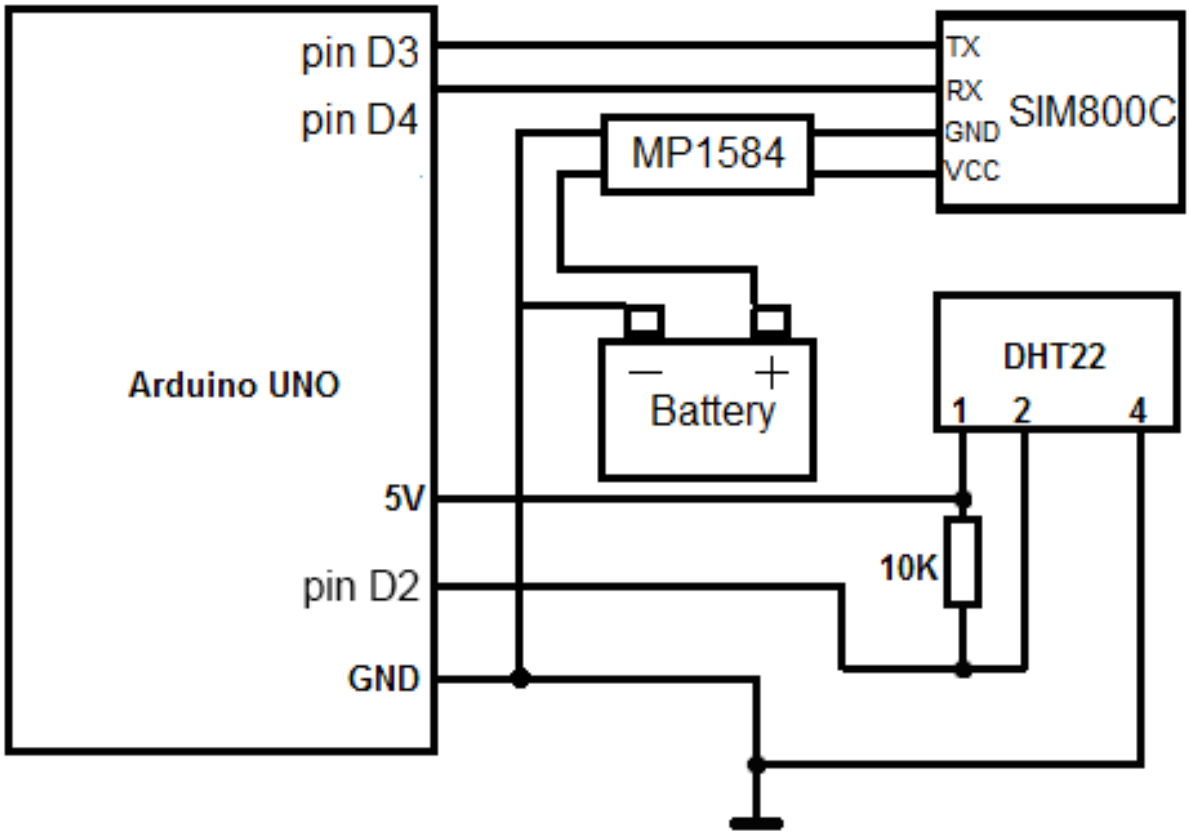
17. Cervo A. Arduino: A Quick-Start Guide. 2nd ed. New York: Pragmatic Bookshelf, 2015. 230 p.

18. Пархоменко С. Мікроконтролерні системи на платформі Arduino: навчальний посібник. Харків: ХНУРЕ, 2021. 156 с.

19. Солдатенко О. Основи робототехніки з Arduino. Київ: Кондор, 2018. 180 с.

20. Hossain S. Arduino and Scilab Based Projects. Singapore: Springer, 2016. 250 p.

ДОДАТОК А. ЕЛЕКТРИЧНА ПРИНЦИПОВА СХЕМА СИСТЕМИ.



ДОДАТОК Б. ВИСХІДНІ КОДИ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

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

SoftwareSerial mySerial(3, 4); // 3=ArduinoRX or SIM800TX,
4=ArduinoTX or SIM800RX
unsigned long delaytime = 300000;//затримка 5 хвилин
DHT   (2,DHT22);
int lowT=5,highT=15;
int lowH=50,highH=80;
int sID=1; //ідентифікатор, номер датчику
int numnormmeas=0; //кількість підряд зафіксованих нормальних
вимірювань (тобто і температура і вологість під час такого
вимірювання у межах норми)
int numconsalar=0; //кількість підряд зафіксованих порушень
int wasalarm=0; //прапорець, чи було порушення параметрів під час
поточного вимірювання

void setup()
{
  // put your setup code here, to run once:
  dht.begin();
  delay(2000);  //час на ініціалізацію модуля GSM
  Serial.begin(9600);  //швидкість порту для видачі діагностичної
інформації при підключенні пристрою до ПК
  Serial.println("Temperature and Humidity Monitoring System v1.0");
  mySerial.begin(9600);
  mySerial.println("AT+CMGF=1");  //режим кодування СМС - звичайний
(для англ. літер)
  delay(100);
  mySerial.println("AT+CSCS=\"GSM\"");  //режим кодування тексту
  delay(100);
}

void loop() {
  // put your main code here, to run repeatedly:
  wasalarm=0;
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  if (isnan(humidity) || isnan(temperature))   // Перевірка. якщо не
вдасться зчитати дані, подається сигнал що датчик вийшов з ладу і
програма зависає на 8 годин
  {
    Serial.println("SMS on sensor broken started");
    mySerial.println("AT+CMGS=\"" + (String)"+380999999999" + "\"");
//номер телефона відповідальної особи - АДМІНІСТРАТОРА системи, який
може співпадати з користувачем
    delay(500);
```

```
    mySerial.print("Sensor "+(String)sID+" vyshel iz stroya!
Trebuetsa zamena!");
    delay(500);
    mySerial.print((char)26);
    delay(500);
    Serial.println("SMS on sensor broken complete");
    for(int i=0;i<8;i++)
      delay(3600000);//затримка на одну зміну - 8 годин
    return;
  }
  //видача діагностичної інформації в порт - потрібно, якщо Arduino
підключений до компютера через USB в режимі відладки
  Serial.print("Humidity, %: ");
  Serial.println(humidity, 1);
  Serial.print("Temperature, C: ");
  Serial.println(temperature, 1);

  if((temperature<lowT)||(temperature>highT))
  {
    Serial.println("SMS on Temeparature alarm started");
    mySerial.println("AT+CMGS=\"" + (String)"+380999999999" + "\"");
//номер телефона відповідальної особи, користувача
    delay(500);
    mySerial.print("Temperatura na sensore"+(String)sID+" ravna
"+(String)temperature+"grad pri norme ot "+lowT+" do "+highT);
    delay(500);
    mySerial.print((char)26);
    delay(500);
    Serial.println("SMS on Temeparature alarm complete");
    delay(2000);
    wasalarm=1;
  }
  if((humidity<lowH)||(humidity>highH))
  {
    Serial.println("SMS on Humidity alarm started");
    mySerial.println("AT+CMGS=\"" + (String)"+380999999999" + "\"");
//номера телефону відповідальної особи, користувача
    delay(500);
    mySerial.print("Vologist na sensore"+(String)sID+" ravna
"+(String)humidity+"% pri norme ot "+lowH+" do "+highH);
    delay(500);
    mySerial.print((char)26);
    delay(500);
    Serial.println("SMS on Humidity alarm complete");
    delay(2000);
    wasalarm=1;
  }
  if(!wasalarm) //оновити прапорці
  {
    numconsalar=0;
    numnormmeas++;
  }
  else
```

```
  {
    numconsalar++;
    numnormmeas=0;
  }
  if(numconsalar>12)//відправити повідомлення директору що вже
годину існує порушення мікроклімату
  {
    Serial.println("SMS to Director started");
    mySerial.println("AT+CMGS=\"" + (String)"+380666666666" + "\"");
//номер телефона директора
    delay(500);
    mySerial.print("Bilshe godyny na sensore"+(String)sID+"
fiksu'jutsa porushennya mikroklimatu!");
    delay(500);
    mySerial.print((char)26);
    delay(500);
    Serial.println("SMS to Director complete");
    delay(2000);
    numconsalar=0;
  }
  if(numnormmeas>96)//8 годин по 5 хвилин (тобто 12 інтервалів за
годину) = 8*12 = 96
  {
    Serial.println("SMS everything OK started");
    mySerial.println("AT+CMGS=\"" + (String)"+380999999999" + "\"");
//номер телефону відповідальної особи, користувача
    delay(500);
    mySerial.print("Za ostanni 8 godyn temperatura ta vologist na
sensori"+(String)sID+" u mezhah normi");
    delay(500);
    mySerial.print((char)26);
    delay(500);
    Serial.println("SMS everything OK complete");
    delay(2000);
    numnormmeas=0;
  }
  delay(delaytime);
}
```