Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Мікропроцесорна автоматизована система безпечної експлуатації приміщень особливої важливості

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ–20-1

_____/ В. К. Гольчик /

Керівник кваліфікаційної роботи _____/ І. А. Котов /

Завідувач кафедри _____/ А. М. Стрюк /

Кривий Ріг

2024

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А. М. Стрюк

«____» _____ 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ–20-1 Гольчик Владиславу Костянтиновичу

1. Тема: Мікропроцесорна автоматизована система безпечної експлуатації приміщень особливої важливості затверджена наказом по КНУ № 279с від «15» квітня 2024р.

2. Термін подання студентом закінченої роботи: «31» травня 2024р.

3. Вихідні дані по роботі: розроблений комплекс має спростити розробку системи безпечної експлуатації приміщень особливої важливості.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): проаналізувати існуючі на ринку аналогічні програмні рішення, обґрунтувати необхідні функції розроблюваного додатку, спроектувати додаток, розробити програмне забезпечення, здійснити тестування розробленого додатку.

5. Перелік ілюстративного матеріалу: функціональна схема, блок–схема алгоритму, зображення екранних форм додатку.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Розгляд літературних джерел та пошук інтернет-ресурсів з заданої тематики	14.01.24 – 21.01.24
2	Аналіз існуючих методів вирішення проблеми	22.01.24 – 02.02.24
3	Формулювання актуальності роботи і постановка завдань	03.02.24 – 19.02.24
4	Оформлення матеріалів першого розділу роботи	20.02.24 – 02.03.24
5	Створення функціональної системи та алгоритму додатку	03.03.24 – 19.03.24
6	Оформлення матеріалів другого розділу роботи	20.04.24 – 09.04.24
7	Розробка баз даних, інтерфейсу програмного забезпечення, програмних модулів	10.04.24 – 01.05.24
8	Оформлення додатків	02.05.24 – 07.05.24
9	Тестування розробленої програми	08.05.24 – 16.05.24
10	Оформлення пояснювальної записки	17.05.24 – 30.05.24

Дата видачі завдання: «12» січня 2024 р.

Студент: _____ / В. К. Гольчик /

Керівник роботи: _____ / І. А. Котов /

РЕФЕРАТ

АНАЛОГОВИЙ СИГНАЛ, АРДУІНО, БАЗА ДАНИХ, ДАТЧИК, ЕЛЕКТРИЧНИЙ ЛАНЦЮГ, КОМПІЛЯЦІЯ, МІКРОКЛІМАТ, МІКРОПРОЦЕСОР, МОНІТОРИНГ, ПРОГРАМУВАННЯ, РОБОЧИЙ ЦИКЛ

Пояснювальна записка: 90 с., 43 рис., 10 табл., 1 дод., 28 джерел.

Мета кваліфікаційної роботи: розробка комплексу автоматизації системи безпечної експлуатації приміщень особливої важливості.

Об’єкт проектування: комплекс автоматизації системи безпечної експлуатації приміщень особливої важливості на основі програмування мікропроцесора Ардуїно.

У теоретичній частині роботи виконано аналіз існуючих апаратно-програмних програмних рішень. Зазначені переваги використання мікропроцесорів при проектуванні цифрових систем, які замінюють традиційну логіку проєктування автоматизованих систем. Обґрунтовані актуальність роботи, мета. Сформульовані завдання для програмного комплексу автоматизації системи безпечної експлуатації приміщень особливої важливості.

У практичній частині кваліфікаційної роботи розроблені структурні і функціональні схеми, апаратно-програмного комплексу, алгоритми роботі його блоків. Сконструйований користувацький інтерфейс програмного комплексу.

Розроблений апаратно-програмний комплекс автоматизації системи безпечної експлуатації приміщень особливої важливості може бути використаний при проєктуванні і експлуатації як житлових будівель, так і промислових споруд особливої важливості.

ABSTRACT

ANALOG SIGNAL, ARDUINO, DATABASE, SENSOR, ELECTRICAL CIRCUIT, COMPILATION, MICROCLIMATE, MICROPROCESSOR, MONITORING, PROGRAMMING, WORKFLOW

Explanatory note: 90 p., 43 fig., 10 table., 1 app., 28 references.

The aim of the qualification work: development of an automation system for the safe operation of premises of special importance.

Design object: automation system for the safe operation of premises of special importance based on Arduino microprocessor programming.

The theoretical part of the work analyzes the existing hardware and software solutions. The advantages of using microprocessors in the design of digital systems that replace the traditional logic of designing automated systems are noted. The relevance of the work and the purpose are substantiated. The tasks for the software complex for the automation of the system of safe operation of premises of particular importance are formulated.

In the practical part of the qualification work, structural and functional diagrams of the hardware and software complex, algorithms for the operation of its blocks were developed. The user interface of the software system was designed.

The developed hardware and software complex for automation of the system for the safe operation of premises of special importance can be used in the design and operation of both residential buildings and industrial facilities of special importance.

ЗМІСТ

ВСТУП……. 8
1 ПОСТАНОВКА ЗАВДАННЯ РОЗРОБКИ АВТОМАТИЗОВАНОЇ СИСТЕМИ ЕКСПЛУАТАЦІЇ ПРИМІЩЕНЬ ОСОБЛИВОЇ ВАЖЛИВОСТІ……... 10
1.1 Загальний аналіз базових принципів функціонування мікропроцесорних комплексів…….. 10
1.2 Лінійка мікропроцесорів ARDUINO та їх апаратна архітектура…. 16
1.3 Програмне забезпечення мікропроцесору ARDUINO……... 21
1.4 Аналоги систем контролю безпечної експлуатації приміщень особливої важливості……. 25
1.5 Постановка завдання побудови мікропроцесорної автоматизованої системи безпечної експлуатації приміщень особливої важливості……. 30
2 АЛГОРИТМИ І МАТЕМАТИЧНІ МОДЕЛІ СИСТЕМИ БЕЗПЕЧНОЇ ЕКСПЛУАТАЦІЇ ПРИМІЩЕНЬ ОСОБЛИВОЇ ВАЖЛИВОСТІ……... 32
2.1 Структурна модель автоматизованого комплексу…… 32
2.2 Математичні моделі процесів у системі моніторингу……... 36
2.3 Конструювання структурної схеми автоматизованої системи безпечної експлуатації приміщень……... 38
2.4 Структури бази даних системи безпечної експлуатації приміщень 43
2.5 Розробка блок-схем алгоритмів модулів системи автоматизації безпечної експлуатації приміщень……... 47
3 ПРОГРАМНА РОЗРОБКА АВТОМАТИЗОВАНОГО КОМПЛЕКСУ БЕЗПЕЧНОЇ ЕКСПЛУАТАЦІЇ ПРИМІЩЕНЬ ОСОБЛИВОЇ ВАЖЛИВОСТІ……... 52
3.1 Розробка блоків системи контролю і безпечної експлуатації приміщень…… 52

3.2 Збірка підсистеми комплексу датчиків системи безпечної експлуатації приміщень особливої важливості…….. 60
3.3 Програмні модулі системи автоматизованого контролю безпечної експлуатації приміщень особливої важливості…….. 65
3.4 Побудова інтерфейсу користувача системи безпечної експлуатації приміщень……. 69
ВИСНОВКИ…… 74
ПЕРЕЛІК ПОСИЛАНЬ…….. 75
ДОДАТОК А……... 77

ВСТУП

Автоматизація машин і процесів дозволила підвищити продуктивність і якість продукції, а також знизити витрати. Але цього недостатньо, коли продукт виявляється невдалим або його життєвий цикл скорочується в результаті появи продукту-замінника. Традиційна автоматизація не дозволяє поліпшити ситуацію в цьому відношенні. Інформаційні технології призвели до нового підходу, згідно з яким виробництво розглядається як потік продуктів, згідно з яким виробництво розглядається як потік матеріалів, що проходить через виробничу систему і взаємодіє з усіма сферами діяльності компанії.

Це породило концепцію комп'ютерно-інтегрованого виробництва (CIM), яка переслідує наступні цілі: зменшити рівень запасів і контролювати їх в режимі реального часу, зменшити прямі витрати, підвищити продуктивність і контроль якості, контроль якості, підвищити доступність машин за рахунок скорочення часу налагодження, забезпечити швидке впровадження нових продуктів.

Крім того, інтелектуальне керуюче обладнання повинно бути інтегроване в єдину систему, в якій вони повинні обмінюватися інформацією один з одним і з IT-системами в інших підрозділах компанії за допомогою таких засобів, як протокол автоматизації виробництва (PAP) для автоматизації виробництва (MAP), який дозволяє об'єднати різне обладнання в єдине комунікаційне середовище.

Важливою особливістю цього проекту є те, що він присвячений сфері автоматизації безпечної експлуатації приміщень особливої важливості. Хоча основною метою була розробка програмного прототипу, що представляє будинок з компонентами домашньої автоматизації, все ж основним було зрозуміти всі концепції, задіяні в процесі розробки проекту.

Більшість попередніх робіт в цій сфері ґрунтується на досвіді створення одноразових прототипів. Наприклад, кілька прикладів експериментально

перевірених прототипів бездротових мереж датчиків-актуаторів для домашньої автоматизації.

Інші дослідження зосереджуються на більш практичних сферах, таких як архітектура. Наприклад, аналізуються зміни в дизайні, пов'язані з впровадженням технологій домашньої автоматизації; або економічні, в яких аналізуються майбутні зміни, пов'язані з програмуванням нових технологій, через зміну кінцевих користувацьких інтерфейсів, які, знадобляться завдяки впровадженню нових технологій домашньої автоматизації.

Arduino - це безкоштовна апаратна платформа, яка розроблена на платі з мікроконтролером і відкритим середовищем розробки, що полегшує користувачам використання електроніки в мультидисциплінарних проектах. Ця безкоштовна, недорога платформа базується на платі вводу/виводу та середовищі розробки IDE, яке реалізує мову процесорного/провідникового обладнання.

За своєю структурою мікроконтролер на платі запрограмований мовою програмування самого Arduino на основі запису, проекти, запропоновані за допомогою цього обладнання, можуть бути виконані без необхідності підключення плати до ПК, це дає легкість руху для проектів, які цього потребують.

Система Arduino має багато версій, однак, найбільш комерційною є Arduino, використовувана версія залежить від проекту і може бути змінена в залежності від його характеристик, в даній конкретній ситуації було вирішене використовувати Arduino, оскільки ця платформа є найбільш стандартною і найкраще підходить для проекту.

1 ПОСТАНОВКА ЗАВДАННЯ РОЗРОБКИ АВТОМАТИЗОВАНОЇ СИСТЕМИ ЕКСПЛУАТАЦІЇ ПРИМІЩЕНЬ ОСОБЛИВОЇ ВАЖЛИВОСТІ

1.1 Загальний аналіз базових принципів функціонування мікропроцесорних комплексів

Мікропроцесорна система (МПС) – або просто мікропроцесор, це найскладніша центральна інтегральна схема в комп'ютерній системі; за аналогією його часто називають "мозком" комп'ютера. Він відповідає за виконання програм, від операційної системи до користувацьких застосунків; він виконує тільки інструкції, запрограмовані на мові низького рівня, виконуючи прості арифметичні та логічні операції, як-от додавання, віднімання, множення, ділення, двійкова логіка та звернення до пам'яті [1, 2].

Розглянемо основні специфікації сучасних складних мікропроцесорних систем. При цьому, розглядається тема їхньої зовнішньої та внутрішньої архітектури, як ця конструкція впливає на опрацювання даних та управління, чи покращує вона їх, чи лише виконує певні функції.

Мікропроцесорна система – це електронна машина, здатна виконувати дуже швидкі обчислення, підкоряючись дуже специфічним і елементарним інструкціям, що відображають її функціональну та організаційну структуру. Концептуально МПС можна визначити як машину, що складається з вхідних елементів, вихідних елементів, центрального процесора і пам'яті [1 - 3].

Існує кілька критеріїв класифікації архітектури МПС. Один з них - спосіб виконання обчислень, що дає змогу відокремити дві категорії МПС: ті, які виконують обчислення послідовно (до них належить більшість комп'ютерів); ті, які виконують процеси паралельно.

В промисловій електроніці найбільш поширеним типом мікропроцесора є мікроконтролер, який об'єднує процесор, пам'ять, порти і деякі периферійні пристрої в одному кристалі.

Мікропроцесори мають безліч застосувань у двох різних сферах: з одного боку, побудова комп'ютерів, а з іншого - виконання завдань управління та приладобудування, замінюючи звичайну логіку.

У першому випадку (найбільш звичному для неспеціаліста) використовуються сучасні мікропроцесори нового покоління, оптимізовані для обробки даних. Наприклад, MP серії Alpha від Digital або PowerPC від Motorola та IBM використовуються в реалізації потужних комп'ютерів, які називаються робочими станціями, орієнтованими на наукові розрахунки та інженерне проектування. Інші, такі як від Intel, більше орієнтовані на розробку персональних комп'ютерів, продуктивність яких, тим не менш, можна порівняти з продуктивністю великих комп'ютерів кількарічної давнини.

Переваги використання мікропроцесорів при проектуванні цифрових систем, які замінюють традиційну логіку, полягають у наступному:

- зменшення кількості компонентів, що призводить до мініатюризації системи, зниження енергоспоживання та підвищення надійності;
- програмованість, що спрощує проектування і скорочує час розробки;
- зниження вартості, спричинене аспектами, описаними в попередніх пунктах, а також виробництвом серій з багатьох одиниць одного і того ж комп’ютера, оскільки його можна застосовувати для вирішення багатьох різних завдань без необхідності перепрограмування.

Тепер зосередимося на основних ідеях, які будуть визначальними далі. Розглянемо загальну схему промислової електронної системи, завданням якої є моніторинг і керування за допомогою електронних процедур певною зовнішньою системою, наприклад, машиною, процесом або навіть цілим промисловим підприємством.

Для прикладу уявімо, що потрібно контролювати та керувати промисловим верстатом. Необхідну електронну систему можна розділити на три модулі. Перший - це підсистема збору даних, по суті аналогового типу, що складається з: набору датчиків, які визначають стан машини; схеми формування сигналу, яка підсилює і адаптує сигнал для подальшої обробки;

нарешті, каскад АЦП, що відповідає за перетворення аналогових сигналів в цифрові значення.

Другий модуль - це система цифрової обробки, яка відповідає за отримання оцифрованих значень сигналів (а також будь-яких суто цифрових сигналів, які можуть існувати), їх обробку, представлення ззовні або запис, а потім надсилання команд для виконання дій на керованій машині. Нарешті, третій блок - це підсистема виконання, яка відповідає за перетворення цифрових сигналів управління в аналогові значення і вплив на машину [3, 4].

Ця схема, є промисловою електронною системою, насправді являє собою загальну електронну систему, яка може бути застосована в багатьох різних ситуаціях, таких як невеликий електронний термостат, гальмівна система, автомат з продажу безалкогольних напоїв, пральна машина, автоматичне управління фермою або навіть управління цілим промисловим підприємством.

У цих випадках цифрове управління здійснюється за допомогою мікропроцесора, хоча іноді це може бути програмований логічний контролер (ПЛК) або комп'ютер. Наприклад, у випадку з промисловим підприємством, серія програмованих логічних контролерів (ПЛК) збирає сигнали і керує механізмами, які контролюються комп'ютером. ПЛК керують механізмами, які контролюються комп'ютером через послідовні лінії передачі даних.

Крім того, потрібне розуміння найважливіших відмінностей у роботі, побудові, перевагах та недоліках мікроконтролерів та мікропроцесорів. Як вони обробляють інформацію, чому мають таку назву, та основні функції.

Існує кілька відмінностей між мікроконтролером і мікропроцесором, першою і найважливішою з яких є функціональність. Для того, щоб мікропроцесор працював, він повинен бути підключений до пам'яті та пристроїв вводу-виводу (I/O) [5].

Загальна схема мікропроцесорної системи приведена на рис. 1.1.

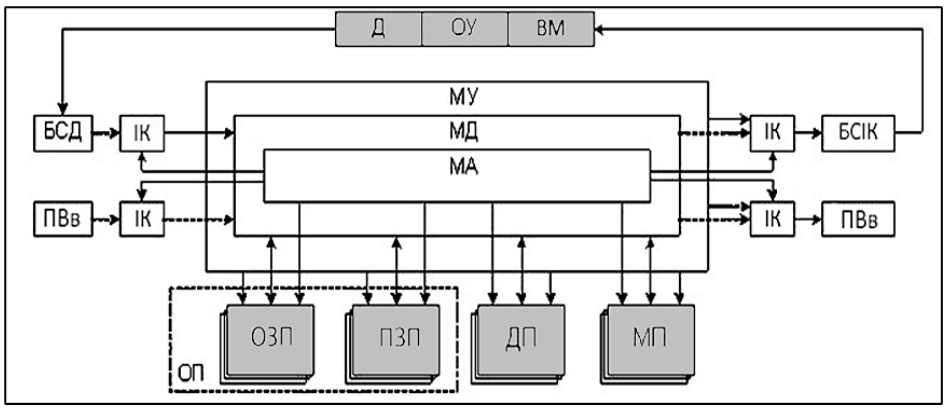

Рисунок 1.1 – Узагальнена схема мікропроцесорної системи керування

На схемі 1.1 показано:

– ОУ – об’єкт керування (управління);
– МУ – магістраль керування (управління);
– МА – лінія (магістральна) адреси;
– БСД – блок з’єднання (сполучення) з датчиками;
– Д – датчики контрольованих величин;
– МД – магістраль контрольованих даних;
– БСІК – блок з’єднання (сполучення) з інформаційними контролерами;
– ОП – загальна (основна) пам’ять;
– МП – мікропроцесор.
– ДП – додана оперативна пам’ять;
– ІК – інформаційні контролери даних;
– ПВ/В – пристрої вводу/виводу;
– ВМ – виконавчі механізми.

SIMD-архітектури відіграють важливу роль у світі паралельних комп'ютерів завдяки своїй здатності обробляти великі вектори та масиви даних за дуже короткий час. Секрет цього типу архітектури полягає в тому, що вони мають кілька процесорів, які виконують одну й ту ж операцію над набором даних. Наприклад, коли одна SIMD-інструкція додає 64 числа, апаратне забезпечення SIMD надсилає 64 потоки даних до 64 ALU (арифметико-логічних блоків), щоб сформувати 64 суми за один такт. Навіть

коли розмір вектору перевищує кількість доступних ALU, швидкість, порівняно з послідовним комп'ютером, є величезною.

Архітектури MIMD, які також називають багатопроцесорними машинами, мають більше одного процесора, що працюють асинхронно і незалежно. При цьому різні процесори можуть виконувати різні інструкції для різних наборів даних. MIMD-архітектури використовуються в системах автоматизованого проектування, комп'ютерній графіці, моделюванні в реальному часі і взагалі в додатках, що вимагають високої обчислювальної потужності [6 - 8].

Алгоритм автомата відіграє дуже важливу роль у проектуванні цифрових систем. Для синхронних схем найкраще підходить техніка діаграм ASM (Algorithm State Machine).

Існують також інші методи, такі як діаграми станів, які є діаграмами, дуже схожими на діаграми ASM. Діаграми станів графічно показують послідовність станів системи, умови кожного стану та переходи між ними. Ця схема не має інших входів, окрім тактового генератора, і інших виходів, окрім тих, що знімаються при кожному перекиданні лічильника.

Стан автомата - це пам'ять про минулу історію, достатня для визначення майбутніх умов. Зазвичай, стан представлено прямокутником з символічною назвою у верхній частині, укладеним у коло.

Рішення дозволяють вибрати шлях, яким повинен рухатися алгоритм відповідно до оціненої вхідної змінної (змінних). Рішення представлені у вигляді ромба з назвою змінної, яка тестується, або функції, яка обчислює кілька змінних.

Безумовні виходи. Використовуються для позначення активації вихідної змінної. Для їх позначення всередині прямокутника стану записуються імена вихідних змінних, які активовані в цьому стані. Необов'язкові виходи не залежать від вхідних умов, вони залежать лише від поточного стану.

Цей тип адресації обмежений для команд з одним записом від кожного

стану. А частина слова пам'яті містить двійкове представлення входу, що тестується у кожному стані, ця частина називається "тестовою частиною". За допомогою цього двійкового представлення селектор входу селектор входу вибирає одну з вхідних змінних. Частина зв'язку має два наступних стани, один з яких обирається селектором зв'язку на основі входу, обраного тестовою частиною.

Для обробки умовних виходів необхідно модифікувати блок-схему адресації входів-станів. Ця модифікація полягає у створенні двох вихідних полів: вихідного поля хибного стану та вихідного поля істинного стану. У першому з них знаходяться виходи, які активуються, коли вхідна змінна дорівнює нулю, а в другому - виходи, які активуються, коли вхідна змінна дорівнює одиниці.

Для проектування модулів керування комп'ютерами потрібні машини станів, які здатні виконувати складніші алгоритми. Шляхом модифікації та додавання компонентів до неявного варіанту адресації можна створити машини станів, які виконують команди з викликом підпрограм, структури DO WHILE, FOR-подібні ітерації та інші.

Розмір слова закодованих інструкцій у мікропроцесорі становить 8 біт. Це число завантажується в регістр команд і розширюється з 8 до 12 біт шляхом додавання чотирьох нулів праворуч. Це нове значення є адресом в мікропрограмній пам'яті, з якої починаються мікрооперації, що виконуються цією командою.

У вихідному полі мікропрограмної пам'яті знаходяться лінії, які керують як внутрішньою, так і зовнішньою архітектурою, які активуються відповідно до виконуваної інструкції.

З іншого боку, входи архітектури вказують на стан як внутрішньої, так і зовнішньої архітектури і дозволяють мікропроцесору приймати рішення відповідно до певних критеріїв.

1.2 Лінійка мікропроцесорів ARDUINO та їх апаратна архітектура

Це безкоштовна апаратна платформа, що базується на платі з мікроконтролером і середовищі розробки, призначена для полегшення використання електроніки в мультидисциплінарних проектах.

Апаратна частина складається з плати з мікроконтролером Atmel AVR та портів вводу/виводу. Найчастіше використовуються мікроконтролери Atmega168, Atmega328, Atmega1280, ATmega8 через їхню простоту та низьку вартість, що дозволяє розробляти різноманітні проекти [9 - 12]. У цьому проекті було використано Atmega328p.

З іншого боку, програмне забезпечення складається з середовища розробки, яке реалізує мову програмування Processing/Wiring, і завантажувача, який виконується на платі.

Arduino може отримувати інформацію з навколишнього середовища через аналогові та цифрові входи, а також керувати світлом, двигунами та іншими виконавчими пристроями. Мікроконтролер на платі Arduino програмується за допомогою мови програмування Arduino (на основі Wiring) та середовища розробки Arduino (на основі Processing). Проекти, створені за допомогою Arduino, можна запускати без необхідності підключення до комп'ютера.

Це безкоштовна апаратна платформа, а це означає, що дизайни доступні на рівні користувача, і ми можемо модифікувати частину з них та адаптувати їх до потреб кожного проекту.

Крім того, існують Arduino Nano, Arduino Mini, Arduino Micro, Arduino Fio [10]. Ці плати менші за розміром і підготовлені для тих випадків, коли проект вимагає невеликого контролера. Це не той випадок, коли розмір не є проблемою.

Загальний вигляд мікроконтролерів ATmega8/48/168/328 показаний на рис. 1.2.

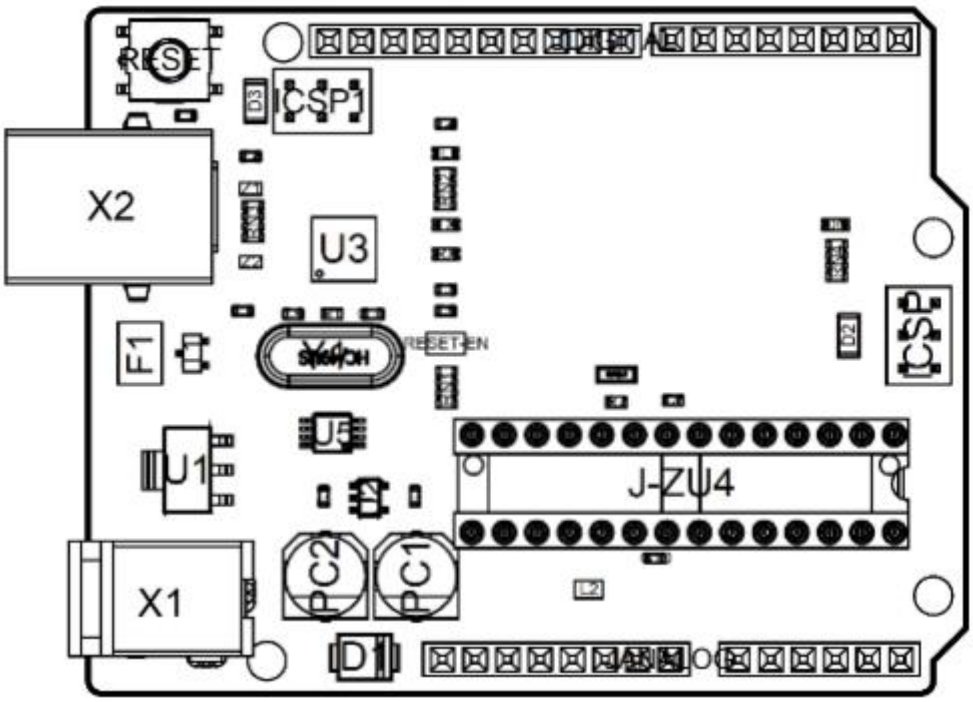

Рисунок 1.2 – Загальний вигляд плати мікроконтролерів ATmega328

Щодо проекту, то Arduino, який можливо використовувати, - це Arduino Mega, розмір коду близький до 32 Кб, тому можливо, що Arduino UNO буде замалим, а входів та виходів, які матимемо, буде недостатньо для Arduino UNO.

Оскільки маємо Arduino UNO для виконання роботи, треба пристосуватися робити систему на цій платі, знаючи, що для її реалізації, оскільки нам знадобиться більше входів/виходів та більший об'єм пам'яті, необхідно використовувати Arduino MEGA [10, 13]. Для реалізації цього коду на Arduino Mega потрібно було б врахувати розводку та деякі конфігурації послідовних портів.

Основні характеристики. – мікроконтролер ATmega328. Робоча напруга - 5V. Вхідна напруга (рекомендована) - 7-12V. Вхідна напруга (обмеження) - 6-20V. Виводи цифрового вводу/виводу - 14 (з яких 6 забезпечують ШІМ-вихід). Аналогові вхідні виводи – 6. Постійний струм на кожний вивід вводу/виводу - 40 мА. Постійний струм на вивід 3,3 В - 50 мА. Флеш-пам'ять - 32 КБ (ATmega328), з яких 0.5 КБ використовується завантажувачем. SRAM - 2 КБ (ATmega328). EEPROM - 1 КБ (ATmega328). Тактова частота - 16 МГц.

Arduino UNO можна живити двома способами: USB-з'єднання (забезпечує 5 В). Замкнемо використовувані послідовні виводи TX, RX COM-порту комп'ютера з виводами Xbee.

Загальна схема мікроконтролера ATmega показана на рис. 1.3.

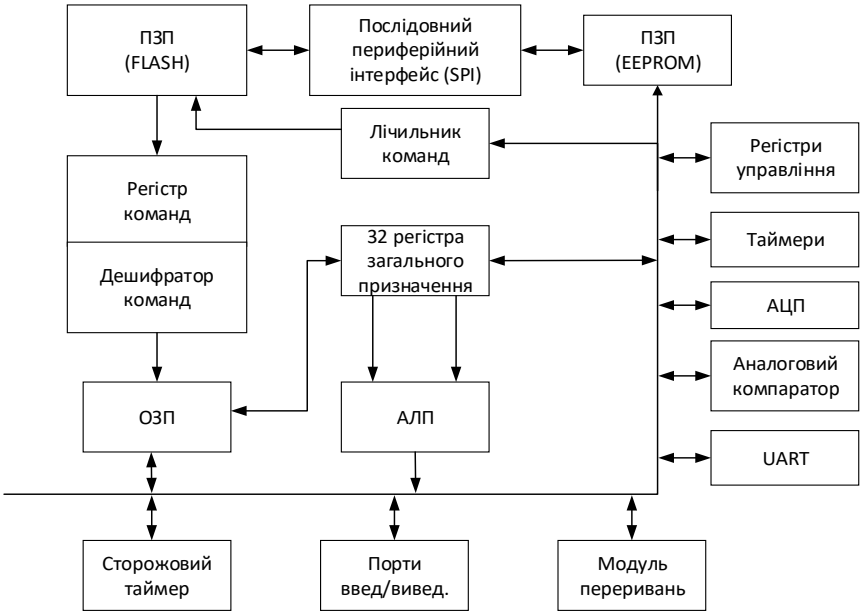

Рисунок 1.3 – Загальна схема мікроконтролера ATmega

Плата UNO є флагманським продуктом Arduino [12, 14]. Незалежно від того, чи ви новачок у світі електроніки, чи будете використовувати Arduino як інструмент для навчальних цілей або промислових завдань, Arduino, швидше за все, задовольнить всі потреби.

Перше знайомство з електронікою. Якщо виконується перший проект в галузі кодування та електроніки, почніть з найбільш використовуваної та задокументованої плати - UNO. Вона оснащена добре відомим процесором ATmega328P, 14 цифровими входами/виходами, 6 аналоговими входами, USB-роз'ємами, заголовком ICSP і кнопкою скидання. Ця плата містить все необхідне для першого досвіду роботи з Arduino.

Плата для розробки за промисловим стандартом. Використовуючи плату UNO R3 в промисловості, є ряд компаній, які використовують плату UNO R3 в якості мозку для своїх систем контролю.

Хоча плата Arduino існує вже більше десяти років, вона все ще широко

використовується в різних освітніх цілях і наукових проектах. Високі стандарти та висока якість виконання плати роблять її чудовим ресурсом для збору даних з датчиків у реальному часі та для запуску складного лабораторного обладнання.

Розглянемо більш детально.

Гніздо живлення (в нашому випадку це 8В). Через клеми Vin і Gnd, таким чином, ми будемо подавати на Arduino напругу живлення 12В. У спроектованій друкованій платі це буде 8В. Якщо два джерела живлення підключені до двох різних каналів, Ардуіно за замовчуванням автоматично вибирає один і ігнорує інші. Вхідні та вихідні значення: в залежності від того, як використовується вивід, ми матимемо наступне. Цифровий вхід і вихід: вихідні значення можуть бути або 0 В (LOW) або 5 В (HIGH), а вхід від 0 до 2 В буде інтерпретуватися як LOW, а від 3 до 5 В як HIGH. Від 3 до 5 В - як високий. Аналоговий вихід: вихідні значення знаходяться в діапазоні від 0 до 5 В з проміжними значеннями від 0 до 255 (8-розрядна точність). Аналоговий вхід: вхідні значення знаходяться в діапазоні від 0 до 5 В з проміжними значеннями від 0 до 1023 (10-розрядна точність). Максимальний струм на всіх цих виводах становить 40 мА. Зазвичай вся електронна схема, якою буде керувати Arduino, монтується на макетній платі, а підключення здійснюється за допомогою кабелів з перемичками (важливо використовувати саме цей тип кабелів, оскільки вони не мають тенденції переламуватися в роз'ємах) [14 - 16].

LCD-дисплей має паралельний інтерфейс, що означає, що мікроконтролер повинен маніпулювати декількома виводами інтерфейсу одночасно, щоб керувати ним. Інтерфейс складається з наступних контактів. Вивід вибору регістра (RS), який контролює, в яку частину пам'яті LCD-дисплея ви записуєте дані. Можна вибрати або регістр даних, в якому зберігається те, що знаходиться на екрані, або регістр інструкцій, в якому драйвер LCD -дисплея шукає інструкції, щоб знати, що робити далі. Вивід читання/запису (R/W), який вибирає режим читання або запису. Вивід дозволу, який активує регістри. 8 виводів даних (D00-D07). Стани цих виводів (високий

або низький) - це біти, які ви записуєте в регістр під час запису, або значення, які ви зчитуєте під час зчитування. Також є вивід контрастності дисплея (Vo), виводи живлення (+5V і GND) і підсвічування (Bklt+ і Bklt-), які дозволяють подавати живлення на РК-дисплей, регулювати контрастність дисплея або вмикати і вимикати підсвічування відповідно.

Що стосується процесу введення та виведення, то мікроконтролер отримує інформацію від датчиків, передає її на виходи, діючи безпосередньо на пристрій, який підключений до нього. Це вимагає підключення мікроконтролера до датчиків на вході і виконавчих механізмів на виході, щоб в залежності від програми і показань датчиків вироблялася серія дій.

Апаратні пристрої, специфікації та принципові схеми яких є загальнодоступними, платними чи безоплатними, називаються вільним апаратним забезпеченням. Філософія вільного програмного забезпечення застосовна і до вільного апаратного забезпечення. Слід завжди пам'ятати, що вільний не є синонімом безкоштовного. Вільне апаратне забезпечення Arduino є частиною вільної культури. Оскільки апаратне забезпечення Arduino пов'язане з прямими змінними витратами, жодне визначення вільного програмного забезпечення не може бути застосоване безпосередньо без модифікацій. Натомість термін "вільне апаратне забезпечення" використовується переважно для відображення використання вільного програмного забезпечення з апаратним забезпеченням і вільного розповсюдження інформації щодо апаратного забезпечення, а також для відображення інформації щодо апаратного забезпечення, часто включаючи випуск принципових схем, проектів і збірок. Плати Arduino повністю відповідають вимогам Регламенту Європейського Союзу (ЄС) 1907/2006 про реєстрацію, оцінку, дозвіл та обмеження використання хімічних речовин (REACH).

1.3 Програмне забезпечення мікропроцесору ARDUINO

Arduino - це відкрита платформа для створення електронних прототипів, що базується на безкоштовному, гнучкому та простому у використанні програмному та апаратному забезпеченні. Вона була створена для художників, дизайнерів, аматорів і всіх, хто зацікавлений у створенні інтерактивних середовищ або об'єктів [5, 12 - 16].

Arduino може отримувати інформацію з навколишнього середовища через свої вхідні виводи, для цього можна використовувати цілий ряд датчиків, і це може впливати на те, що вона є

Arduino може отримувати інформацію з навколишнього середовища через свої вхідні виводи, для цього можна використовувати цілий ряд датчиків і впливати на навколишнє середовище, керуючи світлом, двигунами та іншими виконавчими пристроями. Мікроконтролер на платі Arduino програмується за допомогою мови програмування Arduino (на основі Wiring) та середовища розробки Arduino (на основі Processing). Проекти, створені за допомогою Arduino, можна запускати без необхідності підключення до комп'ютера, хоча вони мають таку можливість і взаємодіють з різними типами програмного забезпечення (наприклад, Flash, Processing, MaxMSP).

Плати можна виготовити власноруч або придбати заводську збірку; програмне забезпечення можна завантажити безкоштовно. Файли CAD доступні під відкритою ліцензією, тому ви можете вільно адаптувати їх до своїх потреб [17].

Програмування - це великий ресурс, який дозволяє нам створювати різні послідовності логічних кроків, які задовольняють наші потреби та потреби наших систем. Програмування - це мистецтво, яке вимагає від програміста великих логічних навичок і концентрації.

Arduino програмується мовою високого рівня C/C++ і зазвичай має наступні компоненти для розробки алгоритму:

– Структури;

- Змінні;
- Математичні, логічні та булеві оператори;
- Керуючі структури (умови та цикли);
- Функції.

Це дві основні функції, які повинна мати кожна програма Arduino:

```
setup(){
}
```

Код початкового налаштування, виконується лише один раз.

```
loop(){
}
```

Ця функція виконується після setup(), вона продовжує виконуватися до тих пір, поки Arduino не буде знеструмлено або від'єднано.

Абревіатура IDE розшифровується як Integrated Development Environment, що в перекладі на нашу мову означає Інтегроване середовище розробки. Це просто спосіб назвати набір програмних інструментів, які дозволяють програмістам розробляти (тобто, в основному, писати і тестувати) власні програми. У випадку з Arduino потрібне IDE, яке дозволяє писати і редагувати програму, яке дозволяє перевірити, чи не допустили ми помилок, і яке також дозволяє, коли ми впевнені, що скетч правильний, зберегти його в пам'яті мікроконтролера плати Arduino, щоб він потім став автономним виконавцем програми [18, 19].

Коли ми говоримо, що мікроконтролер є "програмованим", ми маємо на увазі, що він може постійно зберігати в своїй пам'яті (поки ми не перепишемо її знову, якщо це необхідно) програму, яку ми хочемо, щоб мікроконтролер виконував. Якщо ми не введемо жодної програми в пам'ять мікроконтролера, він не знатиме, що робити. Для того, щоб почати розробку власних скетчів (або спробувати деякі з тих, що є під рукою), нам потрібно буде встановити IDE, що надається проектом Arduino, на наш комп'ютер. Для цього ми виконаємо один з кроків, показаних нижче, в залежності від кожного конкретного випадку.

Рисунок 1.4 представляє інтерфейс Arduino IDE.

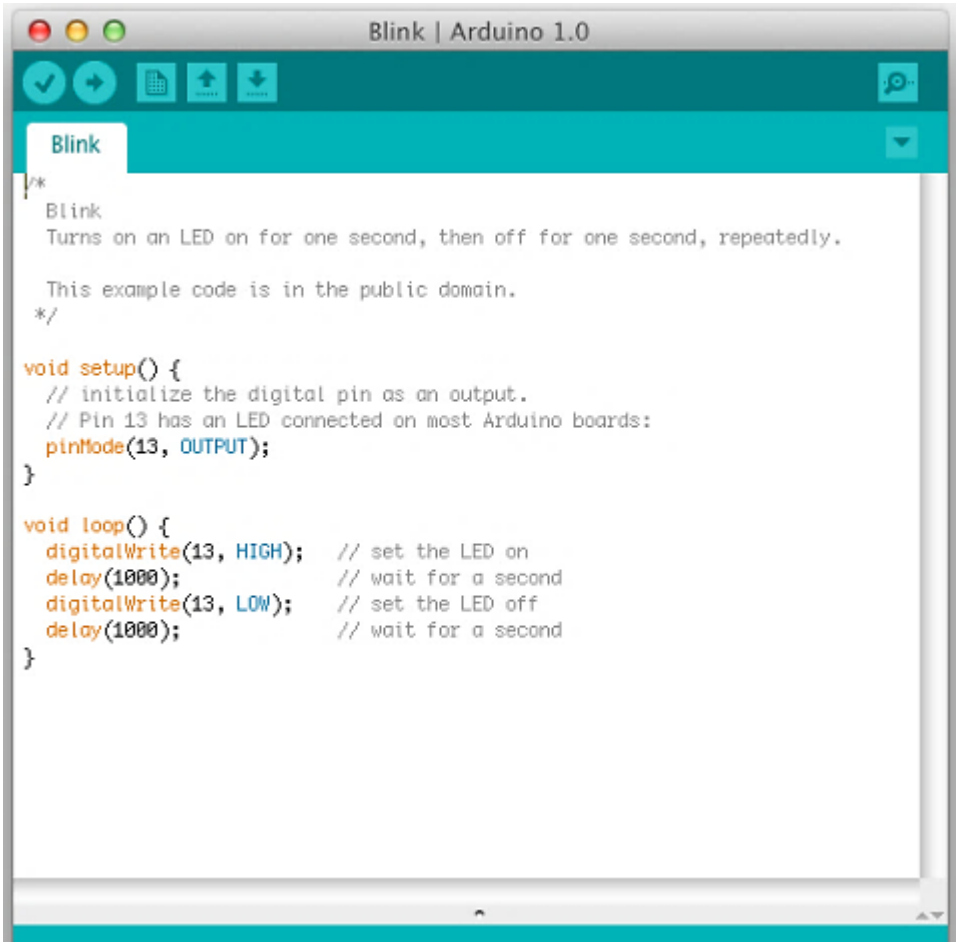

Рисунок 1.4 – Інтерфейс Arduino IDE

Функція - це набір рядків коду, який виконує певне завдання і може повертати значення. Функції можуть приймати параметри, які змінюють їхню роботу. Функції використовуються для розбиття великих проблем на прості завдання і для реалізації операцій, які зазвичай використовуються в програмі, щоб зменшити кількість коду.

Коли викликається функція, керування передається їй одразу після виклику. Коли викликається функція, їй передається керування, а після того, як вона виконає своє завдання, керування повертається в точку, з якої функцію було викликано.

Наприклад:

```
void setup() // Виконується при кожному запуску Arduino
{
  pinMode(13,OUTPUT); // Ініціалізує вивід 13
}
```

```
//-----------------------------------
//Циклічна функція
//-----------------------------------
void loop() // Ця функція продовжує виконуватись
{ // при подачі живлення на Arduino
  digitalWrite(13,HIGH); // Увімкнути світлодіод
  delay(1000); // Таймер на одну секунду (1s = 1000ms)
  digitalWrite(13,LOW); // Вимкнути світлодіод
  delay(1000); // Таймер на одну секунду (1s = 1000ms)
}
```

Щоб встановити Arduino IDE на Windows, перейдіть на її офіційний сайт для завантаження: http://arduino.cc/en/Main/Software. Там, у розділі "Завантаження", ви знайдете посилання на завантаження Windows-версії IDE, яка являє собою не що інше, як стиснутий файл у форматі zip. Тому перше, що потрібно зробити після завантаження - це розпакувати його. Коли ми зайдемо в розпаковану папку, ми не знайдемо ніякого інсталятора або чогось подібного (тільки структуру файлів і підпапок, яку нам не доведеться змінювати), тому що наша IDE готова до використання з цього самого моменту: нам просто потрібно натиснути на виконуваний файл "arduino.exe", і він з'явиться перед нами. Ми можемо перемістити і зберегти папку з файлами, які складають IDE, куди завгодно на нашому жорсткому диску, якщо не будемо нічого змінювати всередині неї.

Рядок меню містить п'ять основних пунктів: "Файл", "Редагувати", "Ескіз", "Інструменти" та "Довідка". Слід зазначити, що ці назви, як і всі елементи IDE, ймовірно, будуть перекладені на мову за замовчуванням вашої операційної системи, але щоб уникнути можливих невідповідностей, у цій книзі ми надалі посилатимемося на їхні оригінальні англійські назви. Пункти рядка меню показують додаткові дії, які доповнюють ті, що можна виконати за допомогою панелі кнопок. Цікаво відзначити, що меню є контекстно-залежним, тобто лише ті пункти, які є актуальними в даний момент (залежно

від того, що ми робимо), будуть доступними, а ті, які не є актуальними, будуть відключені.

Існує багато налаштувань, які не відображаються у спливаючому вікні. Якщо ми захочемо їх змінити, нам доведеться зробити це "вручну", відредагувавши відповідне значення в конфігураційному файлі "preferences.txt", який є нічим іншим, як текстовим файлом, що містить досить зрозумілий список пар даних<->значення. Залежно від вашої операційної системи, файл "preferences.txt" буде знаходитися в іншій папці, яка вказана в самому спливаючому вікні. У будь-якому випадку, цей файл потрібно змінювати, коли IDE не запущено, оскільки інакше внесені зміни будуть перезаписані самим середовищем при його закритті [18, 19].

1.4 Аналоги систем контролю безпечної експлуатації приміщень особливої важливості

Комп'ютери пропонують чудові можливості, які зменшують роботу, яку потрібно виконати для отримання великих переваг. У цьому сценарії архітектура не залишається осторонь, оскільки ці досягнення були прийняті в будівлях для досягнення більшої ефективності процесів, від вертикальних транспортних систем до безпеки самої будівлі. В даний час автоматизація будинків, підприємств, будівель і офісів є тенденцією, що набуває все більшого значення. Кількість людей і компаній, зацікавлених у цій темі, зросла, і вони виявляють високу зацікавленість у впровадженні цих технологічних досягнень у своїх установах; через економію енергії, комфорт, безпеку, ефективний, чіткий і швидкий зв'язок і можливість подовження терміну експлуатації будівлі.

Сьогодні необхідно масово поширювати знання та подальше впровадження домашньої автоматизації, щоб досягти вищої якості життя, комфорту, серед іншого. Розвиток цих технологій сприяє створенню

економічних рішень, які дозволяють замінити імпорт в умовах економічних труднощів.

В багатьох роботах розроблено прототипи розумного будинку, з метою оцінки можливості реалізації його функцій. Крім того, його створення сприяє та заохочує інтерес до вивчення робототехніки.

Поняття, пов'язані з автоматизацією безпечної експлуатації приміщень особливої важливості.

У сфері автоматизації існує різна термінологія, в основному орієнтована на тип програми, що розробляється. Для проекту використовується наступна термінологія.

Автоматизація будівель. Цей термін стосується технічного управління будівлями, а отже, орієнтований на великі компанії, що займаються нерухомістю: готелі, мерії тощо. На відміну від домашньої автоматизації, автоматизація будівель охоплює більші будівлі, з різними конкретними цілями і орієнтовані не тільки на якість життя, але й на якість роботи.

Інтелектуальні будівлі. Домотехнічна будівля, яка включає в себе штучний інтелект, щоб спростити обслуговування, зробити її відмовостійкою, серед іншого. Визначення охоплює й інші сфери, такі як взаємодія з користувачем (стійка та екологічна будівля), інтелектуальне управління інформацією, інтеграція з навколишнім середовищем, простота взаємодії з мешканцями та передбачення їхніх потреб тощо.

Ступені інтелекту в будівлі. Інтелектуальність будівлі - це міра того, наскільки добре вона відповідає потребам мешканців та управління, а також її здатність поважати навколишнє середовище і адаптуватися до нього. Кваліфікатор "інтелектуальний", пов'язаний з технічної точки зору з обладнанням або системою, передбачає наявність принаймні одного процесорного блоку в цьому обладнанні або системі, так що будівля буде технологічно інтелектуальною, якщо вона включає в свою інфраструктуру процесорні блоки, з'єднані між собою за допомогою відкритої системи електропроводки та комунікаційного обладнання.

На рис. 1.5 показано реалізований проект автоматизації безпечної експлуатації приміщень.

Рисунок 1.5 – Приклад автоматизації безпечної експлуатації приміщень особливої важливості

Що стосується характеристик домашньої автоматизації, то ринок пропонує широкий спектр ресурсів, і дизайнер повинен використовувати варіанти, які найкраще відповідають потребам користувача. Фундаментальними концепціями, пов'язаними з домашньою автоматизацією експлуатації приміщень особливої важливості, є

- інтеграція: системи домашньої автоматизації повинні бути інтегровані через домашню мережу, що дає змогу керувати і контролювати ззовні, а також дистанційно оновлюватись і виявляти аномалії;
- продуктивність за мінливих умов: система повинна бути здатна працювати за несприятливих погодних умов, вібрації, перебоїв в електропостачанні, а також мати можливість задовольняти різні інтерфейси, будь то за віковою класифікацією, рівнем технологічних знань або підтримка людей з особливими потребами;

- пам'ять: важливо, щоб система мала можливість зберігати та запам'ятовувати основні функції у випадку збою або відключення електроенергії, а також зберігати історію функцій, щоб їх можна було перевірити або провести аудит;
- тимчасова інформація: система повинна мати можливість розрізняти часові умови (день або ніч), а також кліматичні сезони;
- проста доступність: система повинна мати спрощений інтерфейс, щоб користувачеві було легше нею користуватися, завжди беручи до уваги різний рівень освіти користувачів та їхню залученість до технологічних інновацій;
- легке перепрограмування: у разі виникнення несправностей або навіть збоїв у роботі, система повинна мати можливість перепрограмувати обладнання, а також налаштувати його. Ці функції повинні бути заздалегідь записані і прості в експлуатації;

На рис. 1.6 представлений користувацький інтерфейс системи, яка розроблена в університеті Барселони в рамках дослідницької роботи «Управління приміщенням за допомогою веб-сервісу».

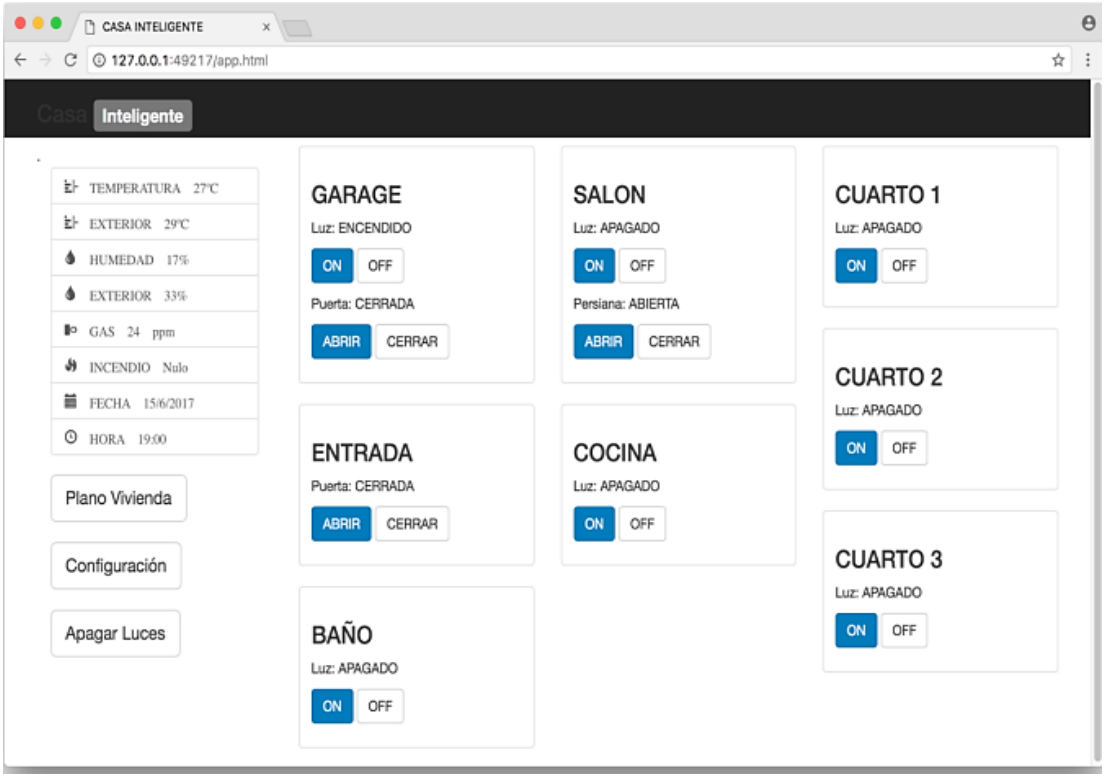

Рисунок 1.6 – Інтерфейс системи контролю приміщення

Розглянемо додатково.

У статтях представлені деякі переваги домашньої автоматизації на світовій арені. Основна увага приділяється безпеці та комфорту цієї нової технології, хоча вона є новою і зростає на ринку. Також йдеться про економію, яку вона приносить користувачеві автоматизованого будинку, а також про деякі аспекти інфраструктури, пояснюючи, як працює ця технологія, дозволяючи бачити і контролювати електричні точки по всьому будинку [20].

Робота трохи розповідає про інтелектуальну домашню автоматизацію, яка являє собою встановлення технологій в будинках з метою поліпшення якості життя, підвищення безпеки та економії електроенергії.

Вона пояснює систему автоматичного навчання, розробку програм, здатних автоматично контролювати ситуації. Він був розроблений, щоб показати, як працює автоматична система автоматизації: як тільки вона отримує команди від датчиків або команди від користувача, програма зчитує їх і надсилає команду для обладнання, щоб виконати завдання.

У статті представлені основні аспекти домашньої автоматизації, включаючи основні елементи системи автоматизації, такі як контролери, датчики, виконавчі механізми та інтерфейси, пояснюється, що вони собою являють і кожна функція цих елементів, показана архітектура системи для роботи автоматизації, сектори, що відповідають за належне функціонування системи, вони поділяються на сектор управління, сектор даних і мультимедійний сектор, додатки для кожного сектора і протоколи зв'язку, за допомогою яких пристрої спілкуються між собою для передачі інформації, виконання команд або візуалізації точок через інтерфейс.

Всі ці проекти розповідають про важливість промислової автоматизації, яка по суті є домашньою автоматизацією, про те, як вона працює і про переваги наявності системи автоматизації у домі [18, 21].

1.5 Постановка завдання побудови мікропроцесорної автоматизованої системи безпечної експлуатації приміщень особливої важливості

Задача, що розглядається в цій роботі, полягає в розробці моделі, що представляє розумний будинок з можливістю реалізації функцій автоматизації безпечної експлуатації приміщень особливої важливості, з деякими автоматизованими компонентами. Ця область відома як домотика, термін який визначає домашню автоматизацію з використанням електронних компонентів, що привносять елементи безпеки, управління, комунікації та оптимізації ресурсів у будинки. Деякі приклади кінцевих функціональних можливостей, які можуть бути реалізовані: автоматична система освітлення, яка вмикається і вимикається, якщо людина присутня, холодильник, який автоматично складає список необхідних продуктів і замовляє їх у постачальників, коли вони закінчуються, або система, яка сповіщає поліцію, коли грабіжник намагається вдертися в будинок.

У кваліфікаційній роботі необхідне реалізувати проект прототипу розумного будинку на основі принципів безпечної експлуатації приміщень особливої важливості, управління яким базується на платформі Arduino. Ця модель має можливість вмикати/вимикати зовнішнє освітлення відповідно до інтенсивності світла. Проєкт повинен мати контроль доступу до освітлення. Внутрішнє освітлення активується за допомогою звукового датчика. Наповнення бака для води автоматизовано за допомогою датчика рівня, який активує водяний насос. Датчик контролює температуру і вологість у приміщенні та активує систему клімат-контролю. Ці функції виконуються автономно. Також є можливість ручного режиму за допомогою додатку. Використовується модуль, який крім запуску всіх вищезазначених функцій, також дозволяє контролювати вікно. Метою роботи є розвиток практичних навичок щодо розробки програмного забезпечення для мікропроцесорних систем на базі мікроконтролеру Ардуіно. Це дозволіть реалізувати додатки для

автоматизації безпечної експлуатації приміщень особливої важливості при відносно низьких витратах і пояснити вплив, який їх використання представляє для безпеки приміщень.

Для вирішення поставленої проблеми в дипломної роботі необхідно забезпечити наступні якості.

Інтеграція. У своїй моделі комп'ютерного управління користувачі не повинні бути присутніми або очікувати на різні події, оскільки вони автономні, з власним програмним дизайном, індикаторами, розташованими в різних місцях з'єднання компонентів від різних виробників.

Взаємозв'язок. Компоненти здатні зв'язуватися в нерівнозначні елементи і таким чином отримувати велику універсальність і різноманітність у прийнятті рішень, як, наприклад, можливість зв'язуватися в роботі кондиціонера з іншими типами побутової техніки, а також з відкриттям вікон або з виявленням користувачів, які знаходяться в їхніх оселях.

Простота використання. Просто глянувши на екран комп'ютера або мобільного пристрою, користувач отримує повну інформацію про стан свого будинку. І якщо він хоче щось змінити, йому потрібно лише натиснути невелику кількість клавіш або просто клацнути мишею, наприклад, температуру всередині і зовні свого будинку.

Пульт дистанційного керування. З тими ж можливостями моніторингу та управління, які доступні локально, ми зможемо отримати через будь-яке інше інтернет-з'єднання з будь-якого іншого компонента і з будь-якого місця, де б ми не знаходилися.

Надійність. Сучасні пристрої дуже потужні, надійні та швидкі. Якщо ми збільшимо використання таких систем, як переривчасте електроживлення, примусова вентиляція процесора, акумулятор великої ємності, який живить периферію, наприклад, автоматичне вимкнення екрану.

Оновлення. Оновити систему дуже просто. Як тільки з'являться нові версії та покращення, нам потрібно буде лише завантажити нове програмне забезпечення на наш комп'ютер.

2 АЛГОРИТМИ І МАТЕМАТИЧНІ МОДЕЛІ СИСТЕМИ БЕЗПЕЧНОЇ ЕКСПЛУАТАЦІЇ ПРИМІЩЕНЬ ОСОБЛИВОЇ ВАЖЛИВОСТІ

2.1 Структурна модель автоматизованого комплексу

Програмна інженерія складається з етапів, що включають методи, інструменти та процедури. Ці етапи складають те, що стало відомо як життєвий цикл програмного проекту. Життєвий цикл являє собою еволюцію системи, продукту, проекту або будь-якого іншого об'єкта, створеного людьми, від його зародження до виходу з експлуатації. Формально життєвий цикл програмного забезпечення можна визначити як різні фази, через які проходить програмне забезпечення від моменту виникнення потреби в механізації програмного проєкту до моменту його виведення з експлуатації. Формально життєвий цикл програмного забезпечення можна визначити як різні фази, через які проходить програмне забезпечення від моменту виникнення потреби в механізації якого-небудь процесу до моменту припинення використання програмного забезпечення.

На рис 2.1 проілюстрована схема програмної системи безпечної експлуатації приміщень особливої важливості.

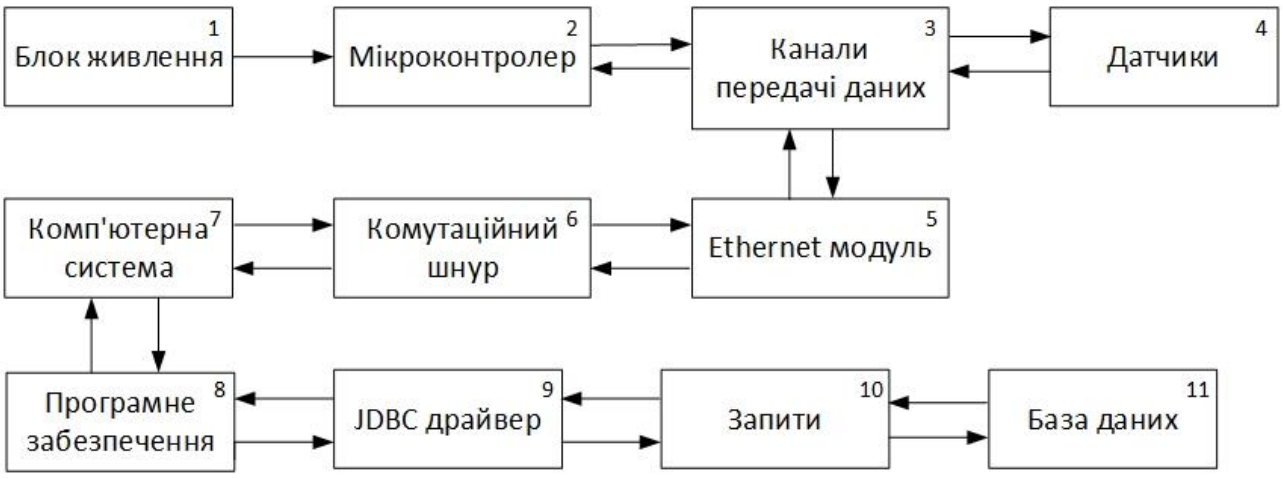

Рисунок 2.1 – Схема програмної системи

Розглянемо цю схему. Підключення плати Arduino до зовнішнього джерела, наприклад до адаптера змінного/постійного струму або акумулятора. У першому випадку на платі є гніздо, куди можна підключити 2,1-міліметровий штекер. У другому випадку дроти, що виходять із клем акумулятора, можна під'єднати до штирьків "Vin" і "Gnd" (позитивний і негативний відповідно) в ділянці плати з написом "POWER". В обох випадках плата теоретично готова до приймання живлення від 6 до 20 вольт, хоча насправді рекомендований діапазон вхідної напруги (з урахуванням бажання домогтися певної електричної стабільності та безпеки в наших схемах) нижчий: від 7 до 12 вольт. У будь-якому разі, ця вхідна напруга, що подається від зовнішнього джерела живлення, завжди знижується до робочої напруги 5 В за допомогою схеми стабілізатора напруги, яка вже вбудована в плату, як показано на рис. 2.2 для плати Atmega328 [8, 22].

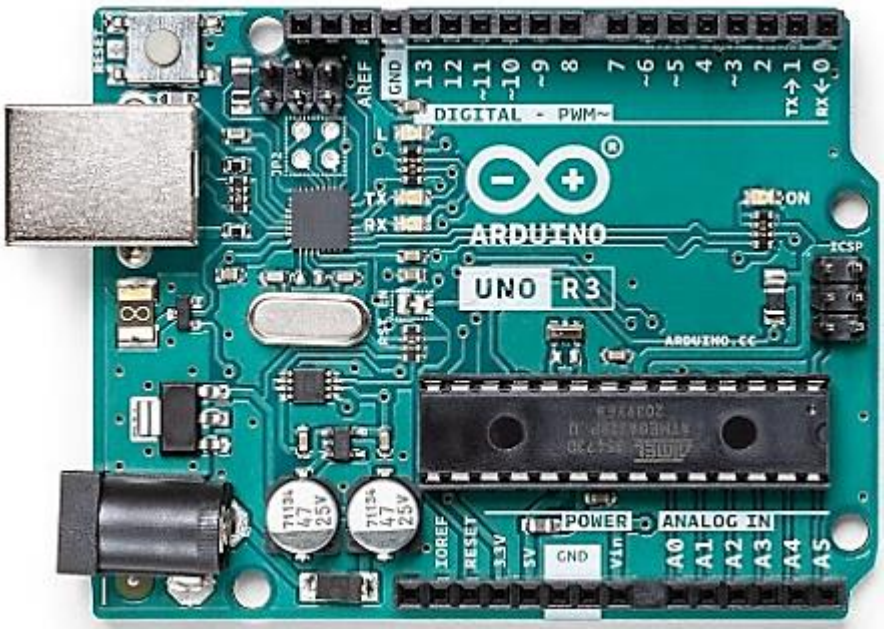

Рисунок 2.2 –Плата Arduino Uno

USB-роз'єм на платі Arduino не лише слугує джерелом живлення, а й є засобом передавання даних між нашим комп'ютером і платою, і навпаки. Цей інформаційний трафік між двома пристроями здійснюється завдяки використанню USB-протоколу - послідовного протоколу, який розуміють і обробляють як наш комп'ютер, так і плата Arduino. На платі необхідно

передбачити елемент "транслятор", який полегшить ATmega328P роботу з інформацією, переданою через USB, без необхідності розбиратися в тонкощах цього протоколу [23]. На рис. 2.3 показаний модуль Arduino Ethernet на платі ENC28J60.

Рисунок 2.3 – Модуль ENC28J60

DHT11 є датчиком вологості, який є комплексом з цифровим сигналом з відповідними показаннями.

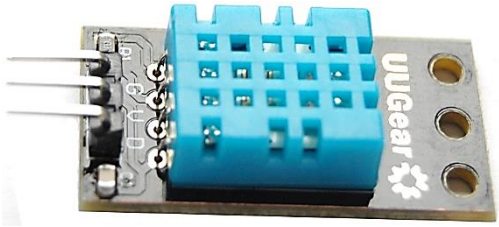

Рисунок 2.4 – Датчик DHT11

У таблиці 2.1 приведені характеристики датчиків:

Таблиця 2.1

Характеристики датчиків

Датчик	Діапазон вологості	Діапазон температур	Напруга живлення
DHT11	20 – 90%	0 – 50°C	5V
DHT22	0 – 100%	-40 – 125°C	3,3 – 6V
SHT1x	0 – 100%	-40 –128°C	3,3V
SHT2x	0 – 100%	-40 – 125°C	3,3 – 5V

Хоча більшість комп'ютерів мають власний внутрішній захист, запобіжник забезпечує додатковий рівень захисту. Якщо на порт USB подається струм понад 500 мА, запобіжник автоматично розриває з'єднання до усунення короткого замикання або перевантаження. На рис. 2.5 показаний датчик витоку води - LM393.

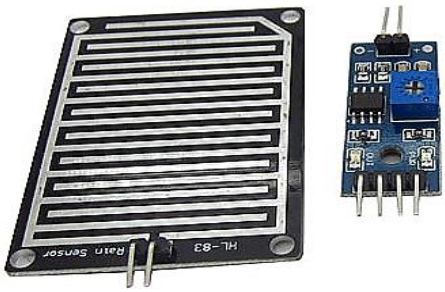

Рисунок 2.5 – Датчик LM393

Далі показаний програмний код Arduino.

```
int sleepModePin = 4; //Де підключено вивід датчика
int motionDetectPin = 2; //Де підключено вивід датчика
int read;
void setup() {
   pinMode(12, OUTPUT);
   pinMode(sleepModePin, OUTPUT);
   pinMode(motionDetectPin, INPUT);
   /*На вивід "sleepModePin" має надходити високий рівень сигналу
 digitalWrite(sleepModePin, HIGH);
}
void loop() {
  read = digitalRead(motionDetectPin);
  if(read == LOW) {
    digitalWrite(12, HIGH);
  } else {
    digitalWrite(12, LOW);
  }
delay(2000);
}
```

2.2 Математичні моделі процесів у системі моніторингу

Закон говорить, що сила струму (I), що протікає через електричний провідник, прямо пропорційна напрузі (V) і обернено пропорційна опору (R).

Формула дуже корисна для того, щоб знати, якій дорівнює змінна, яку ви вказуєте пальцем, наприклад: задайте V (напруга), тоді напруга буде дорівнювати I (струм) помножений на R (опір), ще раз, I (струм), I буде дорівнювати V, поділеному на R.

Розрахунок резистора здійснюється за формулою:

$$R = \frac{(V_S - V_L)}{I}, \tag{2.1}$$

де R – опір резистора (Ом);

V_S – напруга джерела (В);

V_L – напруга світлодіода (В);

I – струм (А).

Параметри RC-кола розраховують наступним чином:

$$F = \frac{1}{(2\pi RC)}, \tag{2.2}$$

де F – частота сигнала;

R – активний опір;

C – ємність кола.

Це фізична властивість, завдяки якій всі матеріали мають тенденцію протистояти потоку струму. Одиницею вимірювання цього параметра є Ом (Ω). Ви можете знайти резистори в електричних нагрівачах, електронних платах, плитах. Вони дуже корисні для обмеження потоку струму. Потік електронів через провідник або напівпровідник в одному напрямку.

На рис. 2.6 приведений приклад дільника напруги.

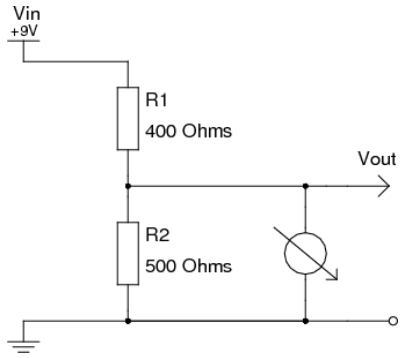

Рисунок 2.6 – Схема дільника напруги схеми

Загальний опір резисторів:

$$R_t = R_1 + R_2 = 300 + 400 = 700\text{Ом} . \quad (2.3)$$

Сила струму через резистори:

$$I = \frac{U}{R_t} = \frac{9}{900} = 0{,}01 = 10mA \quad (2.4)$$

Коли струм в R_2 є відомий, визначимо напругу:

$$V_{out} = I \cdot R_2 = 0{,}01 \cdot 500 = 5\text{В} \quad (2.5)$$

Формула для дільника напруги:

$$V_{out} = V \frac{R_2}{R_1+R_2} . \quad (2.6)$$

Сигнали можуть бути двох типів, описаних нижче: цифрові та аналогові. Також називаються дискретними змінними. Вони характеризуються наявністю двох різних станів і називаються бінарними.

На рис. 2.7 показаний графік навантаження за цикл процесу, який складається з двох ступенів.

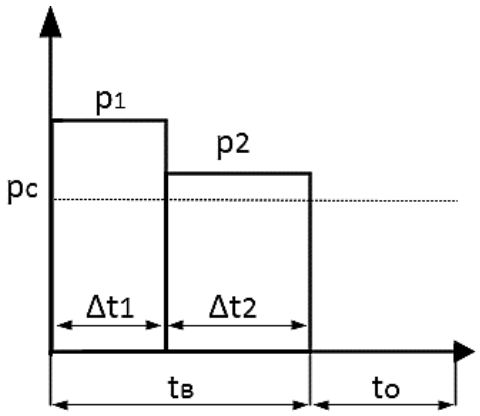

Рисунок 2.7 – Графік навантаження

Формула циклу включає час вимкнення $t_о$, час включення $t_в$

$$t_ц = t_в + t_о \ . \quad (2.7)$$

Час включення циклу обчислюється за формулою

$$t_в = \Delta t_1 + \Delta t_2 \quad (2.8)$$

Обсяг електричної енергії за цикл

$$W_ц = p_1 \Delta t_1 + p_2 \Delta t_2 \ . \quad (2.9)$$

Середня потужність за технологічний цикл

$$p_c = \frac{w_ц}{t_ц} . \quad (2.10)$$

За виразом (2.11) можна підрахувати витрати енергії

$$W_ц = p_с t_с \quad (2.11)$$

Коефіцієнт використання обчислюється за наступним виразом

$$k_в = \frac{w_ц}{p_{ном} \cdot t_ц} = \frac{w_ц}{w_{ном}} \quad (2.12)$$

Розрахунок навантаження для групи приміщень визначається за виразом 2.13.

$$p_р = k_р \sum_i^n k_{иi} p_{ном} \quad (2.13)$$

Для кожного приміщення необхідне обчислити значення коефіцієнтів використання $k_и$ і потужності $\cos\varphi$.

Середньозважений коефіцієнт обчислюється за формулою

$$k_и = \frac{\sum_i^n k_{иi} p_{ном}}{\sum_i^n p_{ном}} \quad (2.14)$$

2.3 Конструювання структурної схеми автоматизованої системи безпечної експлуатації приміщень

Неможливо написати програму безпосередньо в машинному коді: саме для цього існують компілятори. Але, крім того, ці інструменти пропонують ще одну перевагу: необхідно знати, що машинний код, який підходить для одного мікроконтролера, не підходить для іншого, через їх різну внутрішню електронну конструкцію. Тому одну й ту саму програму доведеться кодувати

різними машинними кодами для різних моделей мікроконтролерів. Однак, якщо у нас є спеціальні компілятори для кожної з цих моделей, з одного і того ж вихідного коду ми можемо отримати різні машинні коди, використовуючи відповідний компілятор в кожному конкретному випадку. Рис. 2.8 показує структуру програмного забезпечення [25].

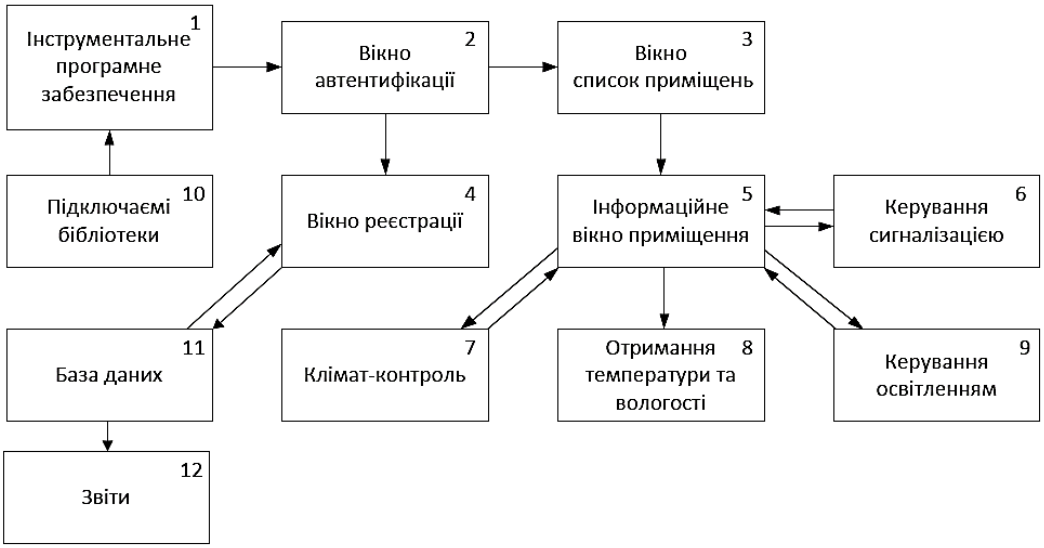

Рисунок 2.8 – Загальна схема програмного забезпечення

Одного разу ми вводимо функцію void loop(){}, яка буде виконуватися циклічно до тих пір, поки ми її десь не зупинимо. Ця функція буде повторюватися нескінченно.

Глобальна змінна є тією, що вказує на хід виконання програми.

На другому екрані ми обираємо ручний/автоматичний режим. Коли ми повертаємо потенціометр, дисплей покаже нам значення, в якому ми перебуваємо. Як тільки ми хочемо отримати це значення (ручний/автоматичний), ми натискаємо кнопку. Кнопка є цифровим входом (5В/0В). Зчитуючи цей вхід, ми дізнаємося, чи він знаходиться на високому рівні (5В) або на низькому рівні (0В). Як тільки він знаходиться на високому рівні, ми виходимо з меню, таким чином збільшуючи змінну на одну одиницю і переходимо на наступний екран.

На рис 2.9 покажемо діаграму класів. Пакет (package) це спосіб, який організовує класи в один простір імен.

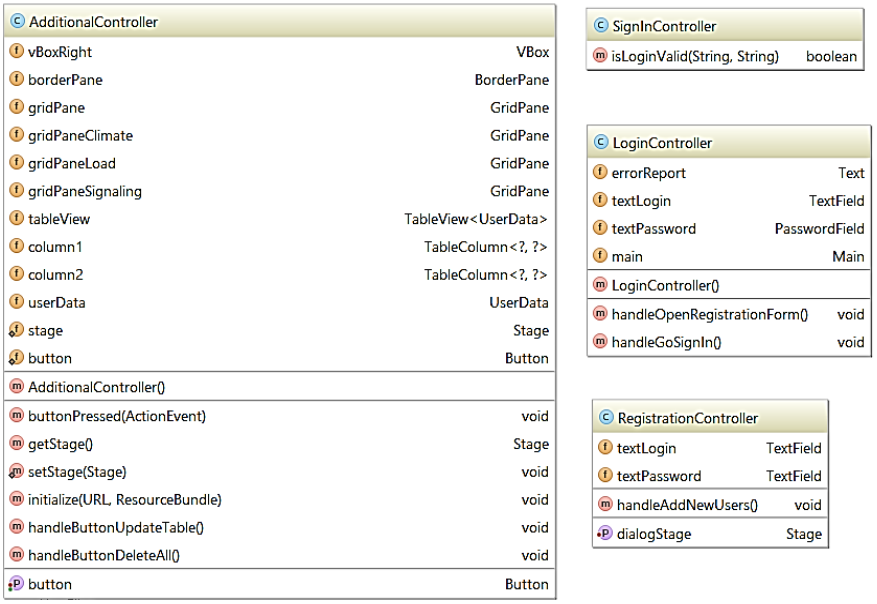

Рисунок 2.9 – Діаграма класів одного пакету

Для здійснення діяльності, якої вимагають програмні процеси, інженери-програмісти повинні застосовувати наукові знання, надані комп'ютерними науками, наукою управління та іншими фундаментальними науками, такими як логіка і математика. Ці знання застосовуються через інженерні процеси та підходи до вирішення проблем, наприклад, системні підходи та системне мислення, що характеризують системну інженерію. Застосовуючи концепції, принципи, методи і техніки менеджменту, інженери-програмісти можуть контролювати три основні змінні інженерії: вартість, час і якість рішення.

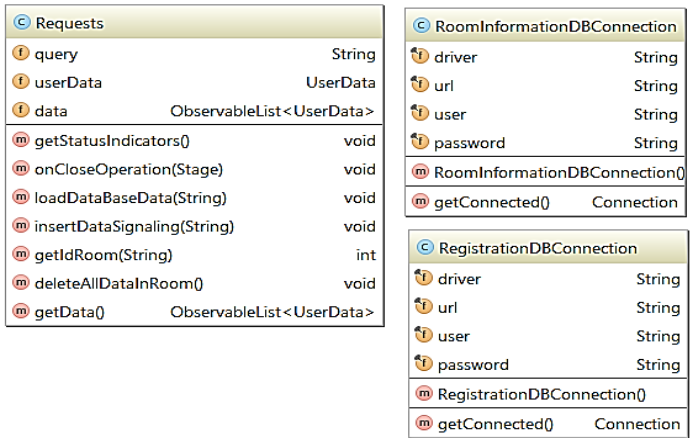

Рисунок 2.10 – Діаграма класів пакета для бази даних

Покажемо діаграму пакету для індикації (рис 2.11)

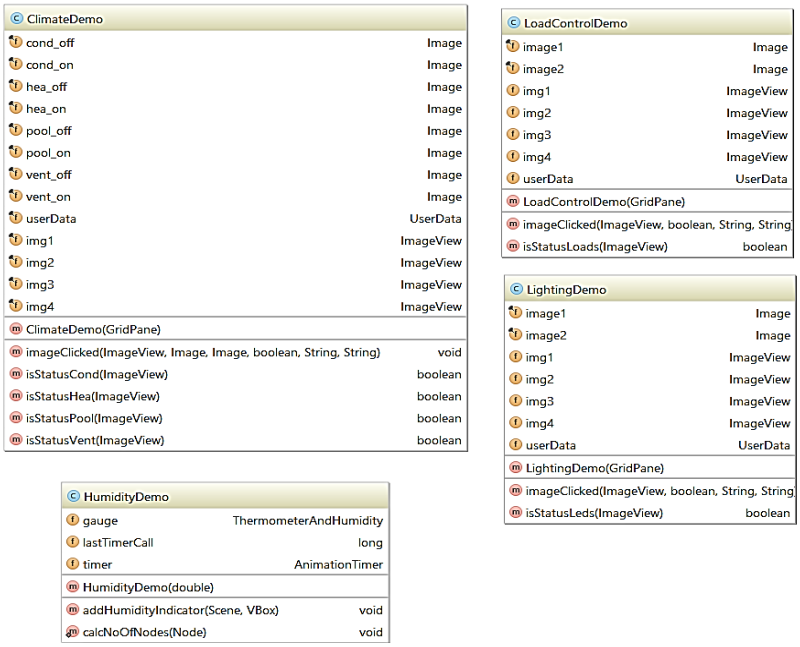

Рисунок 2.11 – Діаграма класів пакету індикації

Композиція представляє ієрархічний зв'язок між об'єктом та його складовими частинами, але в більш сильній формі. Композиція - це ієрархічний зв'язок між об'єктом та його складовими частинами, але в більш сильній формі. У цьому випадку елементи, що входять до складу, не мають сенсу існування, коли перший не існує. коли перший не існує. Тобто, коли зникає елемент, який містить у собі зникає елемент, який містить інші, вони повинні зникнути всі, оскільки не мають сенсу самі по собі, а залежать від елемента, який містить інші. Самі по собі, але залежать від елемента, з якого вони складаються. Компоненти не поділяються між кількома елементами, це ще одна відмінність від агрегації.

Діаграма класів пакета main приведена на рис 2.12. Для цього пакету характерно наступне. Класи керують "одиницею роботи" від початку до кінця. Вони призначені для керування 1) створенням або оновленням об'єктів сутності, 2) екземплярами граничних об'єктів, коли вони отримують інформацію від об'єктів сутності, 3) складним зв'язком між наборами об'єктів і 4) перевіркою даних, що передаються між об'єктами.

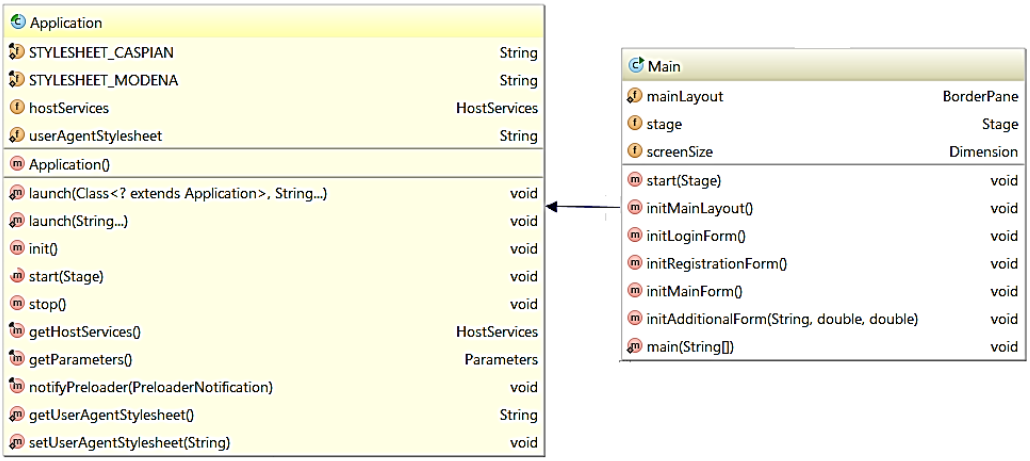

Рисунок 2.12 – Схема класів пакету main

Кожна програма містить певний рівень інтелекту, тобто те, що система знає і що вона може робити. Цей інтелект розподіляється між класами по-різному. Прості класи (ті, що мають мало обов'язків) можуть бути змодельовані так, щоб діяти як підлеглі певних "розумних" класів (тих, що мають багато обов'язків). Хоча такий підхід робить потік управління в системі простим, він має деякі недоліки: він зосереджує весь інтелект в декількох класах, що ускладнює внесення змін, і, як правило, вимагає більшої кількості класів, а отже, і більшого обсягу роботи з розробки.

Якщо інтелект системи більш рівномірно розподілений між класами програми, кожен об'єкт знає щось, робить лише кілька речей (які зазвичай добре ідентифіковані), і згуртованість системи покращується. Це полегшує підтримку програмного забезпечення і зменшує ефект побічних ефектів змін. Використовуючи цей підхід, реалізована схема клієнт-серверної мережевої взаємодії. На рис 2.13 показана схема класів для побудови клієнтського мережевого сокету.

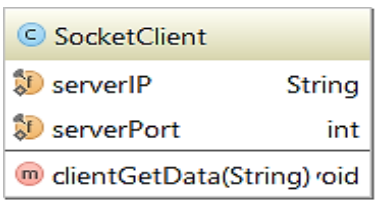

Рисунок 2.13 – Схема класу клієнтського мережевого сокету

2.4 Структури бази даних системи безпечної експлуатації приміщень

Щоб розробити функціональний додаток Java, створимо контролер запитів до сервера баз даних MySQL і спробуємо зробити його об'єктно-орієнтованим [24, 26]. Візуально неважко помітити, що знадобиться щонайменше два вікна: для автентифікації користувача та для написання запитів і відображення їх результатів. З цієї точки зору, можемо визначити чотири області застосування:

- з'єднання з даними;
- аутентифікація користувача;
- SQL запити;
- відображення результатів.

Таким чином, ми інкапсулювали те, що можна назвати мета-об'єктами додатку. Це означає, що кожен з перерахованих вище елементів повинен бути структурований принаймні одним класом, або декількома у співпраці.

Підключення до будь-якого сервера баз даних вимагає наявності додатку, що виконує роль зв'язку. Такі класи називаються конекторами. У випадку з MySQL його потрібно завантажити з сайту. У цьому завантаженні отримуємо весь вихідний код необхідних класів, але потрібен лише файл mysql-connector-java.jar.

Тепер отримаємо підключення до MySQL [27]. Механізм наступний:

- за допомогою методу ForName() класу Class створюється асоціація між нашим класом Connector і драйвером з'єднання MySQL. Якщо це не вдається, то помилка перехоплюється виключенням ClassNotFoundException$;
- з'єднання здійснюється за допомогою методу getConnection() DriverManager; якщо сервер недоступний або відхиляє запит, з'єднання не завершується і генерується виключення SQLException$;

- метод giveMeConnection() відповідає за надання з'єднання класам, які його потребують, а метод closeConnection() скасовує його. Виконання об'єктів бібліотеки sql вимагає виконання всередині блоку try...catch, тому це єдиний рядок, необхідний для розриву з'єднання;
- звичайно, ви повинні мати доступ до бази даних сервера MySQL з ім'ям користувача та паролем. Ці чотири дані: база даних, сервер, користувач і пароль, отримуються як параметр конструктором і потім використовуються методом getConnetion() класу DriverManager.

На рис. 2.14 детально приведено загальна структура JDBC.

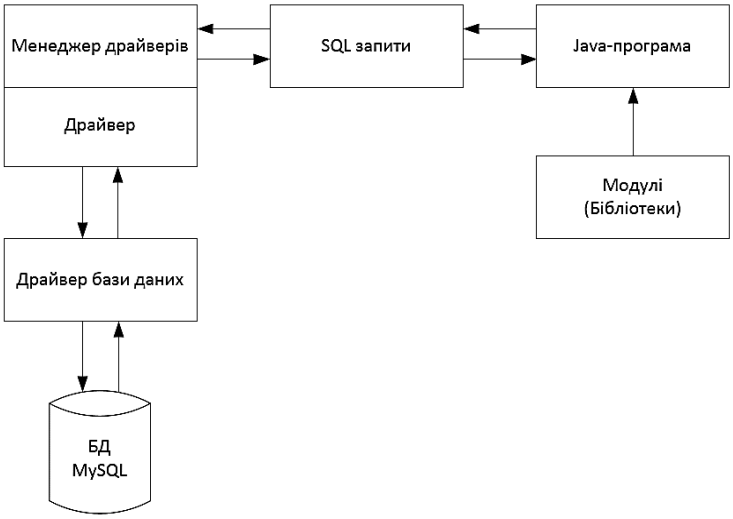

Рисунок 2.14 – Структура JavaDBC

У нашому випадку необхідно додати суб'єкт будинок, ідентифікований кодом (основний ідентифікаційний атрибут), але не забуваючи про те, що номер будинка також може ідентифікувати суб'єкта (альтернативний ідентифікаційний атрибут). Адреса будинка повинна бути представлена як однозначний, обов'язковий і неідентифікуючий атрибут суб'єкта. Крім того, між сутностями будинок також має бути доданий взаємозв'язок. Цей зв'язок зберігатиме інформацію про те, яка служба відповідає за управління орендою цього житла. Буде враховано, що будинок може не відповідати вимогам безпеки як і жодний об'єкт нерухомості в даний момент, але кожним об'єктом нерухомості завжди має керувати одна і тільки одна служба безпеки.

У таблиці 2.2 надана структура всіх таблиць системи безпечної експлуатації приміщень особливої важливості.

Таблиця 2.2

Структура основних таблиць системи безпечної експлуатації приміщень особливої важливості

Ім’я таблиці	Ім’я поля	Тип	Розмір	Індекс (ключ)
1	2	3	4	5
Реєстрація	Login	varchar	20	
	password	varchar	20	
Кімнати	id_room	Лічильник, int	Довге ціле	Первинний ключ
	name_room	varchar	20	
Інформація	id_info	Лічильник, int	Довге ціле	Первинний ключ
	led1	byte	1	
	Led2	byte	1	
	Cond	byte	1	
	Hea	byte	1	
	Pool	byte	1	
	Vent	byte	1	
	load1	byte	1	
	load2	byte	1	
	Fire	byte	1	
	Smoke	byte	1	
	Sign	byte	1	
	Water	byte	1	
	Zam	byte	1	
	room_id_room	int	Довге ціле	Вторинний ключ
Детальна інформація	id_detailed_info	Лічильник, int	Довге ціле	Первинний ключ
	Date	varchar	20	
	situation	varchar	20	
	ind_fire	byte	1	
	ind_smoke	byte	1	
	ind_sign	byte	1	
	ind_zam	byte	1	
	room_id_room	int	Довге ціле	Вторинний ключ

На рис. 2.15 проілюстровані Зв’язки між таблицями MySQL.

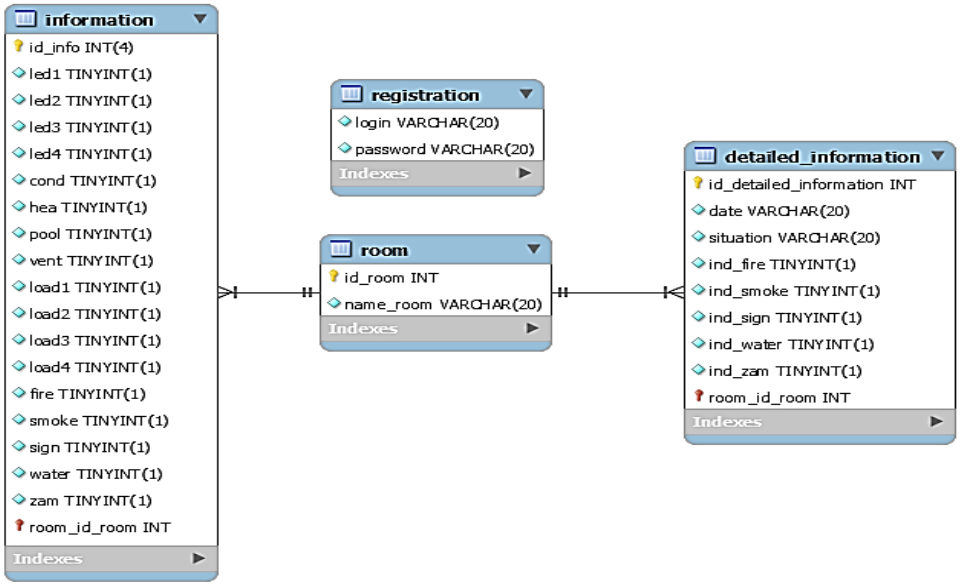

Рисунок 2.15 – Зв’язки у базі даних

Відносно засобів розробки баз даних для платформи Java можна сказати, що це вільна, об'єктно-орієнтована, реляційна система керування базами даних, опублікована під ліцензією. Об'єктно-орієнтована реляційна система управління базами даних. Як і багато інших проектів з відкритим вихідним кодом, розробка не керується компанією та/або окремою особою, а ведеться спільнотою розробників спільнотою розробників, які працюють безкорисливо, безкоштовно та/або за підтримки комерційних організацій, безкоштовних та/або підтримуваних комерційними організаціями. MySQL є найпотужнішою системою управління базами даних з відкритим вихідним кодом на ринку і в своїх останніх версіях не має собі рівних серед інших. Вона використовує модель клієнт/сервер і використовує багатопотоковість замість однопотоковості для забезпечення стабільності. Відносно типів даних слід сказати, що по відношенню до атрибутів існує також поняття домену (множина значень, на яких визначається атрибут). Хоча вони можуть бути представлені в явному вигляді на E/R діаграмах, будемо вважати, що домен має ту саму назву, що й атрибут.

2.5 Розробка блок-схем алгоритмів модулів системи автоматизації безпечної експлуатації приміщень

У кроках, яких слід дотримуватися для розробки алгоритму, є в основному два типи елементів, за допомогою яких можна визначити проблему схематично та за допомогою комп'ютерно-орієнтованої нотації, ці інструменти - алгоритми та блок-схеми. Поняття алгоритму є дуже важливим у сфері обчислень, його фактичне значення схоже на рецепт, процес, метод, техніку, процедуру або рутину для виконання діяльності, за винятком того, що алгоритм має дещо іншу конотацію. На рис. 2.16 приведена схема алгоритму автентифікації користувача.

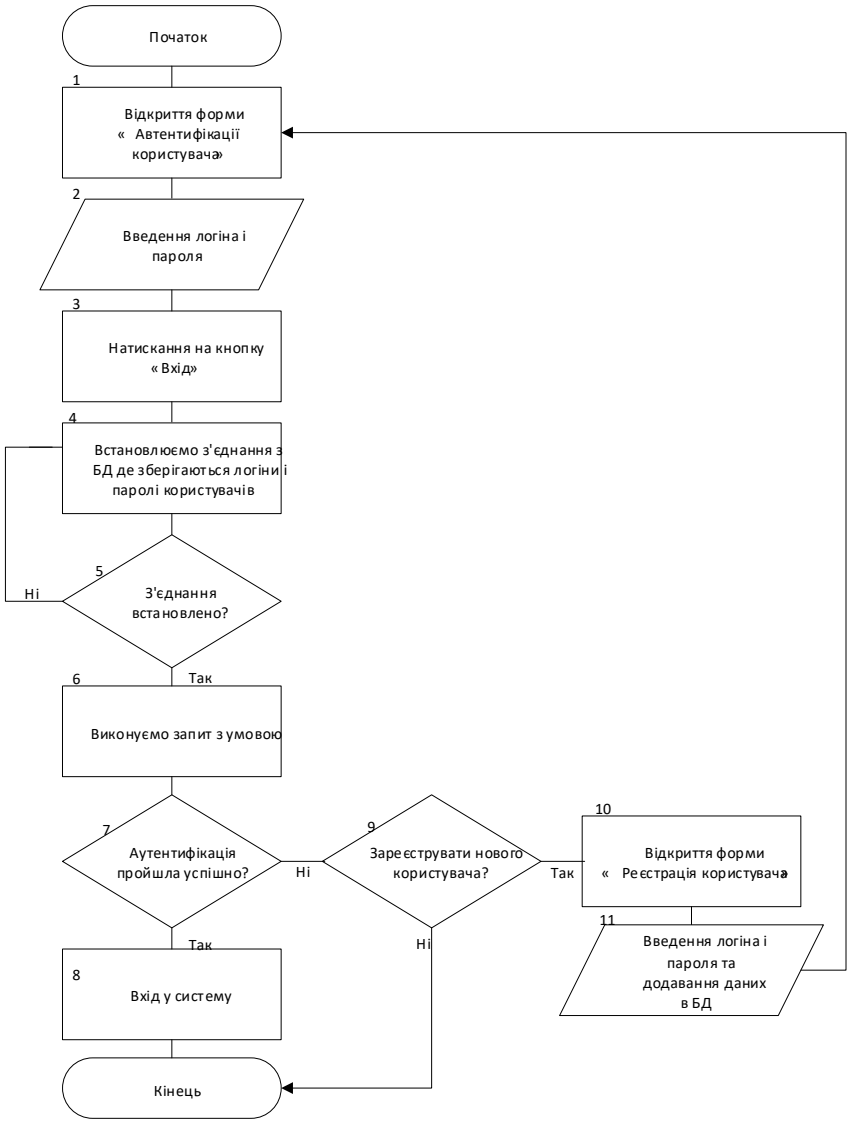

Рисунок 2.16 – Схема алгоритму автентифікації нового користувача

Як бачимо, алгоритм - це набір правил, який відповідає наступним п'яти характеристикам:

- скінченність: алгоритм повинен завершуватися після виконання скінченної кількості кроків;
- визначеність: кожен крок в алгоритмі повинен бути точно визначений, тобто дія, яку потрібно виконати, не повинна бути неоднозначною, а має бути строго визначеною. Алгоритм, описаний такою мовою, як англійська чи іспанська, де одне й те саме слово може мати кілька значень, може не відповідати цьому пункту. Саме тому для специфікації алгоритмів були визначені мови програмування або комп'ютерні мови, оскільки в них значення кожного слова одне і тільки одне;
- вхідні дані: набір даних або інформації, необхідних для розв'язання задачі, вважається вхідними даними. Не будь-який набір даних може розглядатися як вхідні дані в даній процедурі;
- вихід: вихід - це набір результатів, отриманих шляхом застосування алгоритму до набору вхідних даних;
- ефективність: Алгоритм повинен приводити до вирішення поставленої проблеми, іншими словами, можна сказати, що всі операції, що виконуються алгоритмом, повинні бути досить простими, щоб в принципі їх можна було виконати за допомогою паперу і олівця і в підсумку отримати бажаний результат.

Блок-схема складається з набору символів з різними значеннями, які можуть бути з'єднані один з одним. Як і псевдокод, блок-схеми корисні для розробки та представлення алгоритмів, хоча більшість програмістів надають перевагу псевдокоду. У кожній блок-схемі можемо знайти такі елементи: а) початок процесу; б) специфікація потоку даних для виконання процесу; в) дії, що застосовуються до даних; г) отримання результатів; е) завершення процесу. На рис. 2.17 приведено блок-схему алгоритму отримання параметрів вологості і температури у приміщенні.

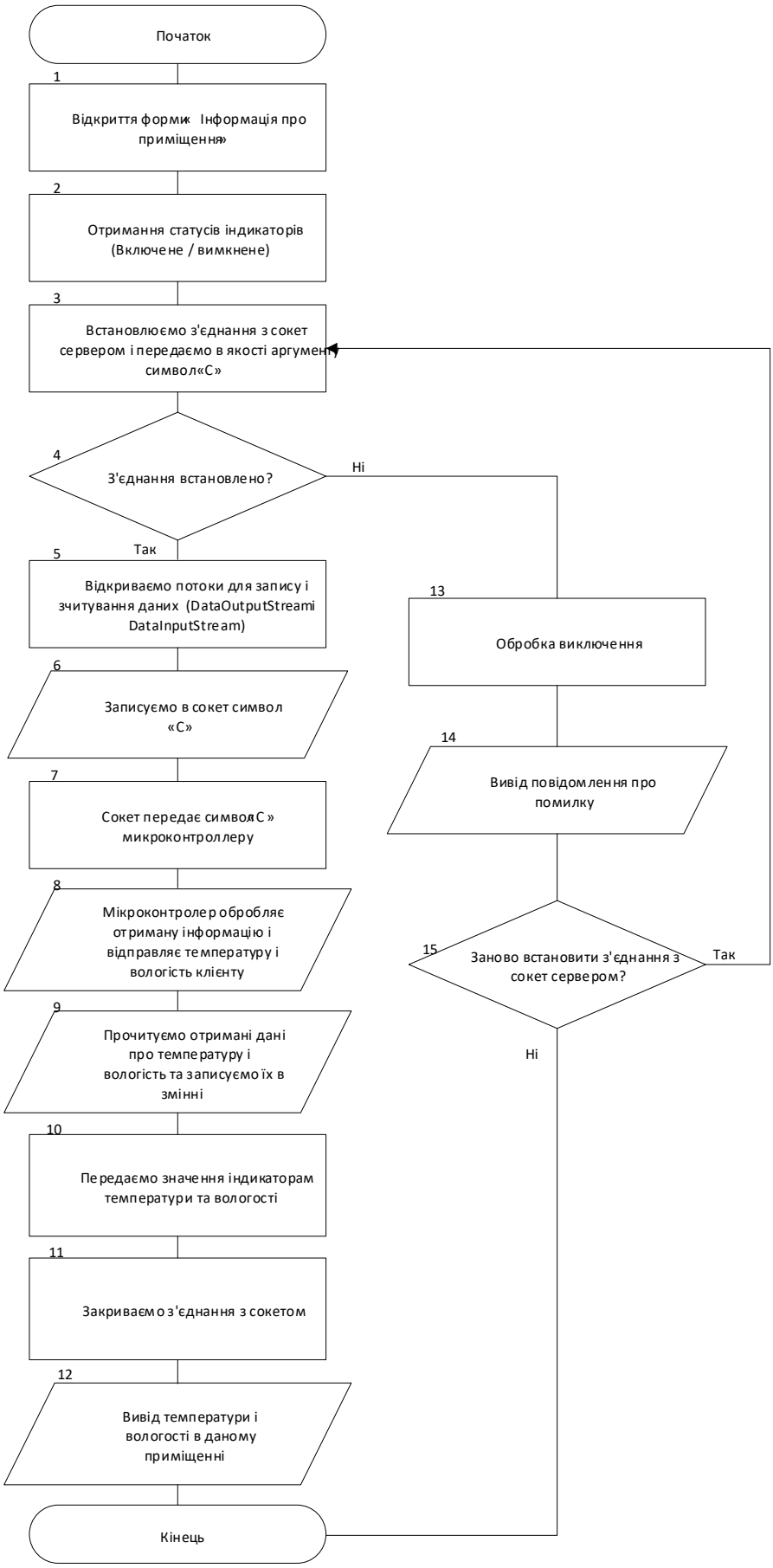

Рисунок 2.17 –Схема алгоритму для отримання кліматичних параметрів в приміщенні особливої важливості

На рис. 2.18 приведена блок-схема алгоритму отримання підключення до бази даних програмного комплексу.

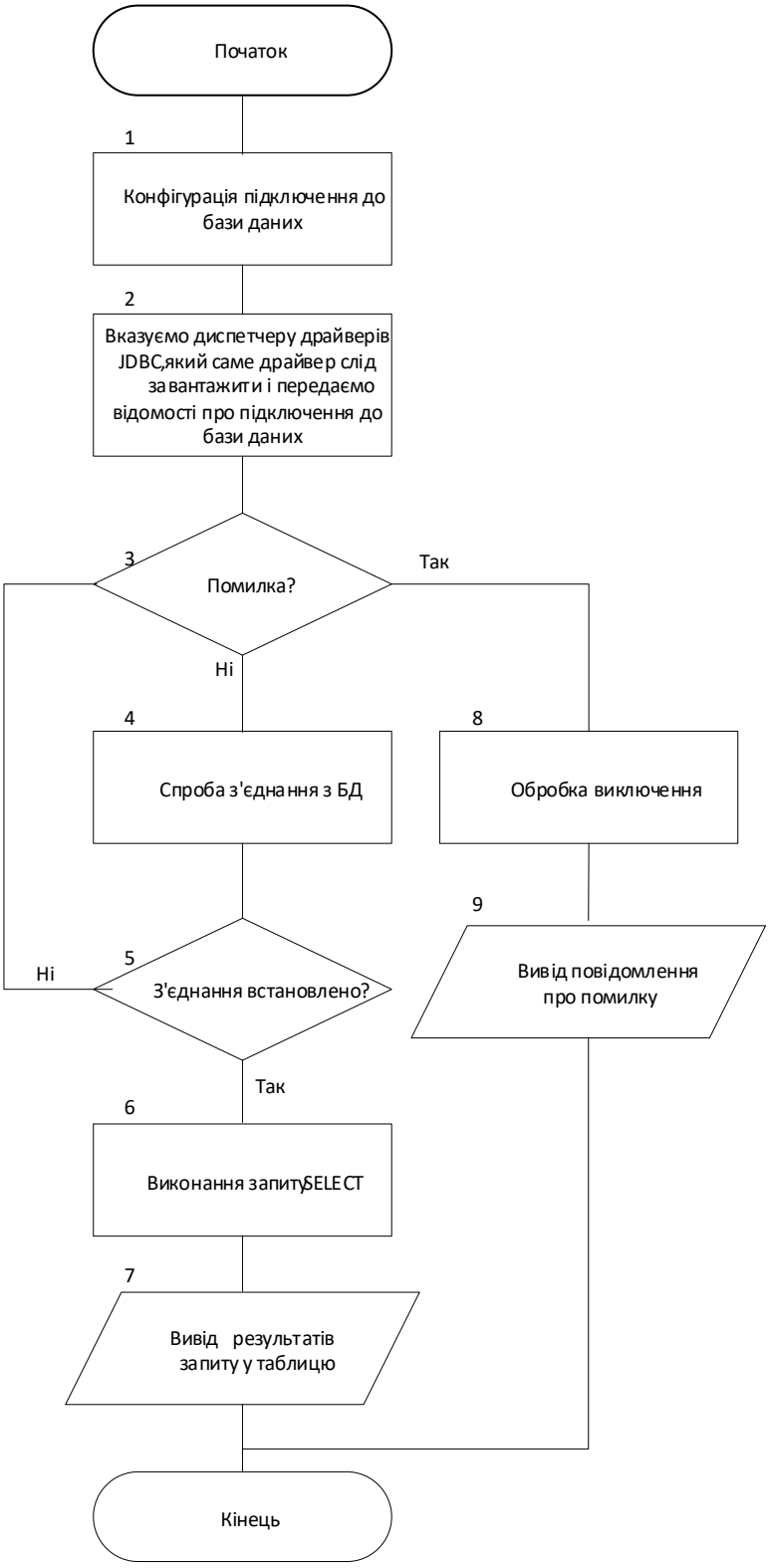

Рисунок 2.18 –Схема програмного алгоритму отримання підключення до бази даних

Насамкінець слід сказати про необхідність колективного розроблення проєктів програмного забезпечення. При цьому можуть виникати різні точки зору та підходи до реалізації проєктів.

Ці протиріччя зазвичай долаються, коли моделі розробляються на різних рівнях деталізації. Там, де існують тісні зв'язки між інженерами по розробці вимог, дизайнерами і програмістами, абстрактні моделі можуть бути все, що потрібно. Конкретні проектні рішення можуть прийматися в процесі реалізації системи, а проблеми вирішуються в ході обґрунтованих дискусій. Якщо зв'язки між специфікаторами системи, дизайнерами і програмістами є непрямими (наприклад, коли система розробляється в одній частині організації, а впроваджується в іншій), можуть знадобитися більш детальні моделі.

Тому важливим кроком у процесі проектування є визначення необхідних моделей та рівня деталізації цих моделей. Це залежить від типу системи, яку потрібно розробити. Система послідовної обробки даних проектується інакше, ніж вбудована система, що працює в режимі реального часу, тому знадобляться різні моделі проектування.

3 ПРОГРАМНА РОЗРОБКА АВТОМАТИЗОВАНОГО КОМПЛЕКСУ БЕЗПЕЧНОЇ ЕКСПЛУАТАЦІЇ ПРИМІЩЕНЬ ОСОБЛИВОЇ ВАЖЛИВОСТІ

3.1 Розробка блоків системи контролю і безпечної експлуатації приміщень

Для Arduino порти можуть мати значення high або low. Процес керування передбачає розміщення даних, які складають зображення того, що ви хочете відобразити, у регістрах даних, а потім розміщення інструкцій у регістрі інструкцій. Бібліотека LiquidCrystal спрощує весь цей процес, тому не потрібно знати низькорівневі інструкції. Hitachi-сумісні системи можуть працювати в двох режимах: 4-бітному або 8-бітному. Для 4-бітового режиму потрібно сім контактів вводу/виводу Arduino, а для 8-бітового - 11 контактів. Для відображення тексту на екрані більшість речей можна робити в 4-бітному режимі. Функції, які виконуватиме система, полягатимуть у зборі даних, надісланих Arduino, зборі даних, введених користувачем, а також у відображенні та доступі до всіх попередніх даних [28]. Він також слугуватиме для перевірки походження даних та верифікації користувачів, щоб ніхто не міг маніпулювати даними, не знаючи кодів доступу. На рис. 3.1 показані діючи порти апаратного комплексу Arduino для підключення датчиків.

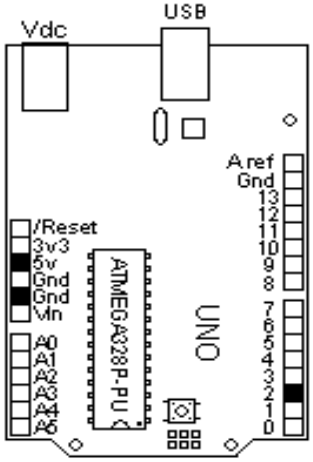

Рисунок 3.1 –Порти Arduino для включення DHT11

За замовчуванням виводи Arduino (Atmega) є вхідними, тому немає необхідності явно конфігурувати їх як входи за допомогою pinMode(). Вважається, що виводи, знаходяться у включеному стані. Це можна пояснити тим, що вхідні клеми мають надзвичайно малі вимоги до схеми, з якої вони здійснюють вибірку, що еквівалентно послідовному резистору номіналом 100 МОм перед виводом. Це означає, що для переходу вхідного виводу з одного стану в інший потрібен дуже малий струм, що дозволяє використовувати виводи для таких завдань, як використання ємнісного сенсорного датчика або зчитування аналогового датчика за схемою RCTime.

Це також означає, що вхідні клеми, до яких нічого не підключено, або з підключеними до них проводами, не підключеними до інших ланцюгів, відображатимуть, здавалося б, випадкові зміни стану виводів, вловлюючи електричні перешкоди з навколишнього середовища або ємнісний зв'язок від стану сусідніх виводів. У нас є кнопка S1 як такий тип цифрового входу. Властивості виводів, сконфігурованих як вихід (OUTPUT). Вважається, що виводи, сконфігуровані як OUTPUT за допомогою pinMode(), перебувають у низькоомному стані. Це означає, що вони можуть пропускати значну кількість струму в інші схеми. Виводи ATmega можуть подавати позитивний струм або негативний струм до 40 мА (міліампер) на інші пристрої або схеми. На рис. 3.2 приведені порти платформи Arduino для включення модуля ENC28J60.

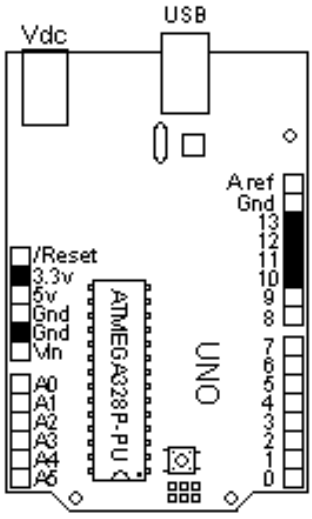

Рисунок 3.2 – Порти Arduino для підключення модуля Ethernet

Підтягуючий резистор становить 10 кОм. Коли перемикач натиснуте, вхід схеми, як і раніше, підключений до дійсного вхідного сигналу. Коли перемикач відпускається, вхід схеми з'єднується зі знижувальним резистором, який опускається на землю (яка завжди є фіксованою точкою відліку).

Коли перемикач натиснуте, схема отримає вхідний сигнал, але також буде з'єднана з землею через тягнучий резистор: що тоді відбувається насправді? Ось ключ до того, чому використовується розтяжний резистор, а не пряме з'єднання з землею: протидія проходженню електронів, що надходять від зовнішнього сигналу, яку чинить розтяжний резистор, призводить до того, що вони завжди будуть перенаправлятися на вхід схеми. Якби ми з'єднали вхід схеми з землею напряму, не використовуючи резистор, зовнішній сигнал був би спрямований безпосередньо на землю, не проходячи через вхід схеми, тому що він зустрічав би менший опір (чистий закон Ома: менший опір, більший струм). За допомогою підтягувального резистора можна було б досягти того ж самого [12, 15]. У цьому випадку, коли перемикач натиснуто, зовнішній сигнал відводиться на землю, оскільки він знаходить прямий шлях до неї (тому на вхід схеми нічого не надходить - "0"), а коли перемикач не натиснуте, на вхід схеми надходить зовнішній сигнал. З цим потрібно бути обережним.

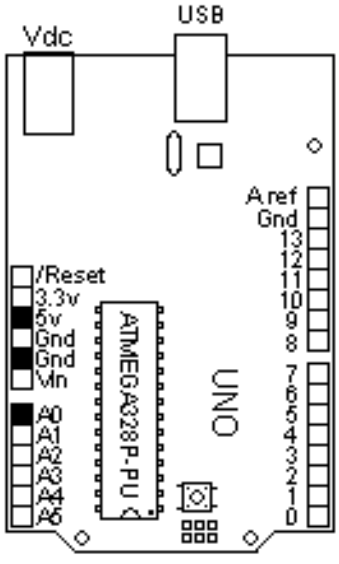

Рисунок 3.3 – Порти Arduino для включення датчика води

Джерелом живлення ми називаємо елемент, який відповідає за створення різниці потенціалів, необхідної для протікання електричного струму по ланцюгу, щоб підключені до нього пристрої могли працювати. Існує два типи джерел живлення, які ми найчастіше будемо використовувати в проектах: акумулятори та AC/DC адаптери.

Термін "батарея" використовується для позначення генераторів електроенергії, заснованих на хімічних процесах, які зазвичай є незворотними і тому не перезаряджаються, тоді як термін "акумулятор" зазвичай застосовується до напівзворотних електрохімічних пристроїв, які можуть перезаряджатися, хоча ці терміни не є суворим формальним визначенням. Термін "акумулятор" взаємозамінно застосовується до обох типів (а також до інших типів генераторів напруги, таких як електричні конденсатори) і тому є нейтральним терміном, здатним охопити і описати їх усі. Найпоширенішими типами є D (LR20), C (LR14), AA (LR06) і (LR03), які генерують 1,5 В і мають циліндричну форму, хоча вони різняться за розміром (фактично, вони перераховані від більшого до меншого). Тип PP3 (6LR61), які генерують 9 В і мають форму прямокутної призми, також поширені. Перші зазвичай випускаються у вигляді тонких прямокутників у срібному футлярі, а другі - у твердому прямокутному або циліндричному корпусі, хоча і ті, і інші відрізняються великою різноманітністю і гнучкістю форм і розмірів. На рис. 3.4 приведені порти платформи Arduino для включення модуля реле.

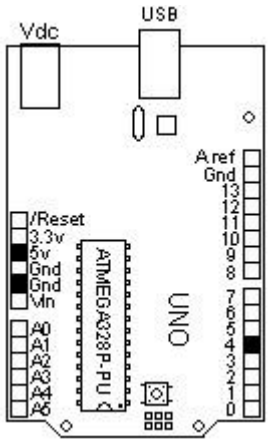

Рисунок 3.4 –Порти Arduino для включення модуля реле

Щоб сигналізація спрацювала при виявленні вторгнення крім датчиків і виконавчих механізмів, з яких складаються системи, їм потрібен центральний блок, здатний обробляти логічний алгоритм, який запускає дії. Arduino має всі ці характеристики. Він має ряд виводів, які можна налаштувати як входи, якщо потрібно зчитувати інформацію з датчиків, або як виходи, якщо потрібно запустити виконавчий механізм. Пояснимо датчики та виконавчі механізми більш детально. Крім того, він має перепрограмований мікропроцесор, на який ми зможемо завантажити або завантажити програму, і він буде відповідати за зчитування інформації з датчиків, підключених до вхідних контактів, і активувати (встановлювати HIGH або 5 вольт) або деактивувати (встановлювати LOW або 0 вольт) вихідні контакти. Іншими словами, ми можемо налаштувати Arduino для створення простої системи безпеки наступним чином. Визначимо Pin2 Arduino як вхід. До цього виводу ми підключимо наш датчик присутності, який завжди знаходиться на 0 вольт, за винятком випадків, коли він виявляє присутність, і тоді він піднімається до 5 вольт. Отже, нам потрібно лише неодноразово (в циклі) зчитувати значення Pin2 Arduino і запитувати, чи є на ньому 5 вольт, або ж він знаходиться у високому стані. Якщо ця умова виконується, це означає, що є відкрите полум'я і, отже, ми повинні активувати сирену, щоб сповістити про це (рис. 3.5).

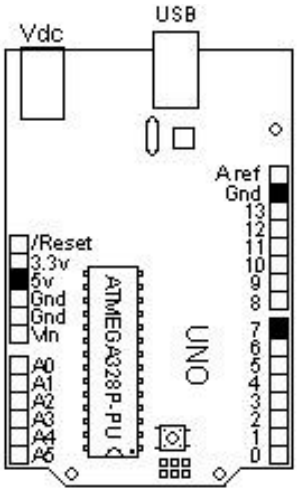

Рисунок 3.5 – Порти Arduino датчика полум'я

Якщо необхідно використовувати гніздові виводи плати Arduino, ми вже бачили, що з попереднім екраном це неможливо. Можливим рішенням є використання окремого модуля з тим самим сенсорним датчиком і мікросхемою контролера, підключеного кабелями до плати Arduino і розміщеного таким чином (наприклад, на макетній платі), щоб невикористані контакти роз'єму Arduino були доступними і ніщо їх не блокувало. Такий модуль існує і є продуктом Adafruit під номером 335. Він працює точно так само, за винятком того, що він не має роз'єму для microSD.

Цей модуль можна підключити до плати Arduino двома різними способами, обидва з яких надруковані на одній стороні. На одному з них зображена смужка з 20 отворами (яку потрібно припаяти до ланцюжка стандартних контактів, оскільки на заводі вони не припаяні), а на іншому - дві смужки з десятьма отворами, але обидва способи еквівалентні. В основному, потрібно підключити три "компоненти" модуля: підсвічування, сам датчик і сенсорний датчик, а також підключення самого датчика (для якого потрібні 2 цифрових і 2 аналогових виводи, але їх можна використовувати повторно, оскільки вони не заважають роботі), а також підключення самого датчика (для якого потрібні 2 цифрових і 2 аналогових виводи, але їх можна використовувати повторно, оскільки вони не заважають роботі). На рис. 3.6 приведені порти платформи Arduino для включення датчика руху.

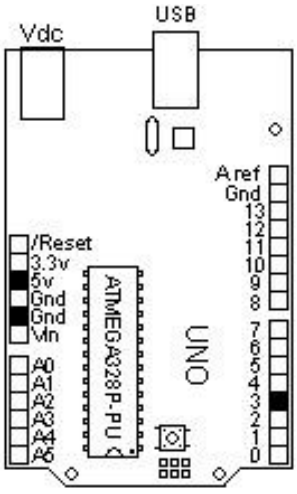

Рисунок 3.6 –Порти Arduino для підключення датчика можливого руху

Роз'єми дещо відрізняються в залежності від виробника, але всі вони, як правило, сумісні для використання на макетних платах тощо. Однак, слід зазначити, що роз'єми, які розповсюджуються в офіційному магазині Arduino - це роз'єми типу Tinkerkit, оскільки вони спеціально розроблені для підключення до Tinkerkit "Sensor Shield" (щит, спеціально обладнаний різними роз'ємами Tinkerkit для підключення різних сумісних датчиків та актуаторів без необхідності використання макетної плати).

Їх робоча напруга становить від 5 до 7 В, тому їх можна живити від самої плати Arduino. Однак, енергоспоживання датчику газу пропорційне концентрації, тим більше енергії він споживає. Це означає, що на практиці, в залежності від навантаження, необхідно буде використовувати додаткове незалежне зовнішнє джерело живлення 5В, щоб забезпечити його окремим живленням від того, що пропонується через плату Arduino (але завжди із загальним заземленням). Додаткове живлення також буде необхідне, коли в наших схемах використовується більше двох датчиків, незалежно від того, яке навантаження вони мають. Перевірити датчик легко: переведіть мультиметр в режим безперервного струму і підключіть будь-який провід від мультиметра (оскільки датчики є неполяризованими пристроями) до кожної клеми датчика. Якщо підключити датчик послідовно зі світлодіодом (і необхідним дільником напруги) і подати живлення, то світлодіод вмикається або вимикається залежно від концентрації, на яку налаштована система. На рис. 3.7 приведені порти платформи Arduino для включення датчика газу.

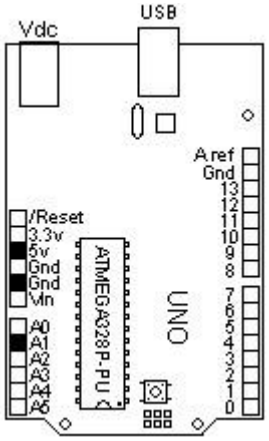

Рисунок 3. 7 –Порти Arduino для датчика газу

Герконовий сенсор завжди містить мікросхему, яка використовується для зчитування вихідного сигналу сенсора і обробки його таким чином, щоб в кінцевому підсумку випромінювати цифровий імпульс назовні (саме його і отримає наша плата Arduino). Більшість герконових датчиків поставляються на платах з 3 контактами з'єднання з одного боку або знизу: живлення, заземлення і сигнал даних. Порядок розташування цих трьох контактів може відрізнятися в залежності від моделі, тому перевірте специфікацію (хоча здебільшого кожен контакт має свою функцію, надруковану на самій платі). Живлення зазвичай становить 3-5 В постійного струму, але може бути і до 12 В, тому з живленням від 5 В до 9 В вони працюють чудово.

Інші характеристики, спільні для більшості моделей, полягають у тому, що при виявленні руху вони видають високий імпульс (3,3 В), а при відсутності руху - низький імпульс. Тривалість імпульсів визначається резисторами та конденсаторами, присутніми на друкованій платі, і відрізняється від датчика до датчика. Діапазон чутливості, як правило, становить до 6 мм, а кут чутливості - 110° по горизонталі і 70° по вертикалі. Більшість моделей містять мікросхему BIS0001 для керування внутрішньою схемою та датчик RE200B. По суті, датчик являє собою контакт, який змінює свій опір залежно від сили магнітного поля, що на нього діє. Ці датчики дуже дешеві і прості у використанні, але не дуже точні: одне й те саме вимірювання може відрізнятися від одного датчика до іншого на 10%. Отже, що можна очікувати – це можливість реагувати на діапазони спрацьовування.

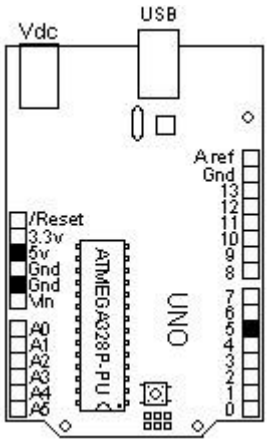

Рисунок 3.8 –Порти Arduino для включення герконового датчика мережі

3.2 Збірка підсистеми комплексу датчиків системи безпечної експлуатації приміщень особливої важливості

Робототехніка пропонує стартовий набір Arduino, який складається з ретельно підібраних найкращих компонентів, щоб ви могли розпочати роботу з Arduino. Це інструмент, який містить все необхідне для розробки програм, що сприяють вивченню та управлінню Arduino.

На рис. 3.9 приведені контакти для включення сенсора DHT11.

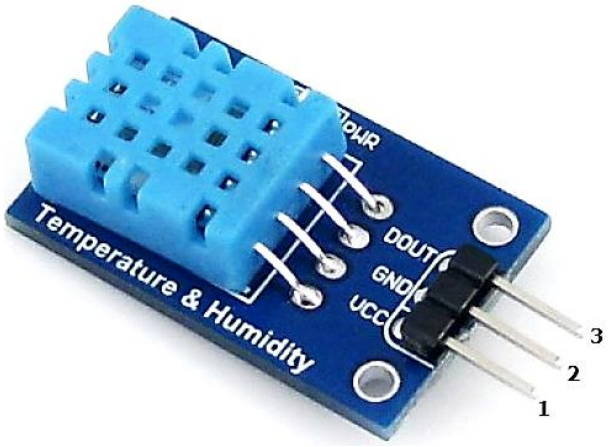

Рисунок 3.9 – Контакти сенсора DHT11

Адаптер Ethernet - це просто невелика плата, що містить роз'єм типу mini-B і мікроконтролер ATmega8U2, мікросхему, дуже схожу на відому ATmega16U2 (але з половиною флеш-пам'яті, звідси і назва), яка запрограмована на виконання точно такої ж функції: перетворення USB-з'єднання в простий послідовний сигнал 5В, зрозумілий мікроконтролеру ATmega328P на платі Ethernet. Тому, щоб підключитися через USB до плати Ethernet, ми повинні спочатку підключити цей адаптер до шести контактів, які виступають з плати Ethernet (кожен з яких підключений безпосередньо до певного контакту на ATmega328P: RX, TX, скидання, живлення, заземлення), а потім підключити USB-кабель до гнізда, передбаченого адаптером. Як і у випадку з платою UNO, новий USB-роз'єм можна використовувати як для живлення плати, так і для її програмування.

На рисунку 3.10 приведені контакти для включення модуля Ethernet на базі контролера ENC28J60.

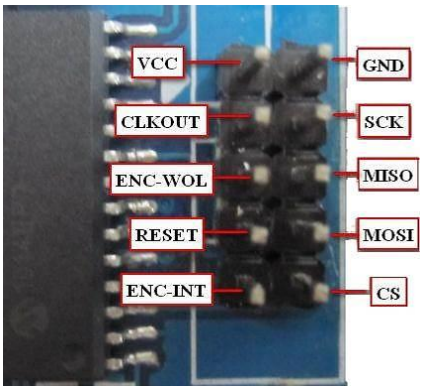

Рисунок 3.10 – Контакти Ethernet-модуля ENC28J60

Датчики витоку води набагато дешевші за інші типи датчиків, вони - це всього лише резистори і працюють при будь-якій напрузі. Їх важко пошкодити через їхню простоту і вони неймовірно точні у своїх вимірюваннях. Наприклад, резистор на 10 кОм (номінальне значення, взяте за стандартну температуру 25 °C) може вимірювати опір з точністю до ±0,25 °C (за умови, що аналого-цифровий перетворювач також є достатньо точним). Однак вони зазвичай не можуть витримувати температуру понад 100 і кілька градусів, а їхня постійна часу (тобто секунди, за які резістор зменшує різницю між початковою і кінцевою температурою на 63 відсотки) зазвичай перевищує 10 секунд. Для вимірювання опору, як і будь-якого іншого резистора, можна використовувати мультиметр. Отримане значення буде залежати від температури місця, де ми знаходимося.

Наступним природним кроком після того, як відомо значення опору контактів, є з'ясування, якій температурі він відповідає. Найпростіший спосіб зробити це - звернутися до таблиці еквівалентності, яка завжди міститься в даташиті, де для певних значень безпосередньо вказана відповідна температура води. Ця система буде добре працювати, якщо все, що ми хочемо - це спроектувати схему, яка робить швидкі порівняння. На рис. 3.11 показані виводи для підключення датчика витоку води на мікросхемі LM393.

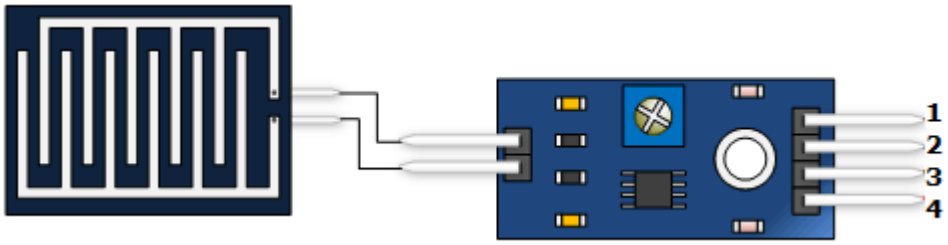

Рисунок 3.11 – Контакти сенсору витоку води

Микроконтроллеры в основном используются для приема сигналов от устройств ввода/вывода, и их преимущество в том, что можно обойтись без внешних схем. Порты ввода-вывода микроконтроллера сгруппированы в порты длиной 8 бит, которые позволяют считывать данные извне или записывать их изнутри микроконтроллера. Целью является работа с простыми устройствами, такими как реле, светодиоды, двигатели, фотоэлементы, кнопки и все остальное, что может придумать программист.

На рис. 3.12 приведені контакти для включення модуля реле.

Рисунок 3.12 – Виводи релейного модуля

Датчики відкритого полум’я зазвичай невеликі, дешеві і прості у використанні, тому їх широко застосовують в побутових пристроях загалом. Але вони неточні: кожен датчик реагує інакше, ніж інший, навіть якщо вони виготовлені в одній партії. Саме тому їх слід використовувати не для визначення точних рівнів інтенсивності полум’я, а скоріше для визначення коливань інтенсивності світла, які можуть походити від самого навколишнього полум’я, або від наявності перешкоди, що блокує прийом падаючого світла. Можна також мати систему фоторезисторів і таким чином порівнювати, який з них отримує більше світла в певний момент. На рис. 3.13 приведені контакти для підключення сенсору відкритого полум’я.

Рисунок 3.13 – Контакти сенсору відкритого полум’я

Інфрачервоний датчик руху включає інфрачервоний світлодіод та інфрачервоний датчик спеціально для пульта дистанційного керування. Також він може компонуватися як датчик освітленості, датчик температури, датчик нахилу, датчик удару (використовується як зумер), датчик магнітного поля (разом з магнітом), датчик сили та акселерометр. Другий - інфрачервоний датчик, специфічний для пульта дистанційного керування, датчик освітленості, датчик вигину, датчик удару і вібрації, датчик магнітного поля (разом з чутливим до нього перемикачем - так званим герконом), датчик сили, датчик вологості, датчик відстані, датчик руху, акселерометр, гіроскоп, компас (магнітометр) і датчик атмосферного тиску.

На рис. 3.14 приведені контакти для включення піроелектричного датчика руху.

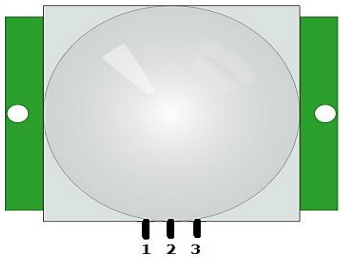

Рисунок 3.14 –Піроелектричний датчик руху

Набір датчиків включає в себе декілька модулів GROVE, серед яких ми можемо знайти датчики газу, а також потенціометри, джойстики, кнопки, світлодіоди, зумери, гучномовці тощо. Він також включає в себе так званий "Grove Base Shield", який є нічим іншим, як щитом, який можна прикріпити до плати Arduino з роз'ємами типу GROVE, куди можна підключити різні модулі GROVE. Для керування модулями GROVE, на відміну від TinkerKit, не

обов'язково використовувати якусь специфічну бібліотеку: з мовою Arduino, як вона є, цього достатньо. На рис. 3.15 приведені контакти для включення датчика газу.

Рисунок 3.15 – Контакти сенсора витоку газу

Технологія THT - це найпростіший спосіб припаяти компонент до друкованої плати. Для цього потрібно, щоб компоненти мали металеві штирі завдовжки кілька міліметрів, а в друкованій платі були просвердлені отвори: технологія полягає в тому, що кожен штир проходить через отвір у платі, а потім припаюється до нижньої сторони друкованої плати (за бажанням можна відрізати міліметри штиря, які можуть виступати з-під нього після паяння). З іншого боку, за технологією SMT компоненти мають не штирі, а невеликі металеві виступи, які припаюються до спеціальних невеликих. Для простоти з’єднання використовуються герконові контакти.

На рис. 3.16 приведена плата для включення герконового датчика.

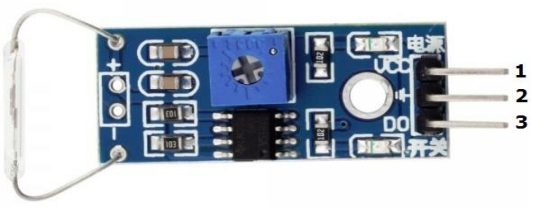

Рисунок 3.16 – Контакти герконового сенсору

Ця плата призначена для стаціонарного встановлення в об'єктах або експонатах. З цієї причини пластикові штифти повинні бути розміщені в отворах "вручну" (або дроти повинні бути припаяні безпосередньо). Це дозволяє використовувати різні типи конфігурацій відповідно до потреб.

3.3 Програмні модулі системи автоматизованого контролю безпечної експлуатації приміщень особливої важливості

Розробимо основні модулі програми для платформи Arduino. Перший розділ (який не має початкового або кінцевого символу-розділювача) зарезервовано для написання, як вказує його назва, різних оголошень змінних, які нам потрібні. У наступному розділі ми детально пояснимо, що все це означає. Всередині двох інших секцій (тобто, всередині їх ключів) ми повинні написати інструкції, які ми хочемо виконати на нашій платі, пам'ятаючи про наступне. Інструкції, записані всередині секції "void setup()", виконуються тільки один раз, в момент включення (або скидання) плати Arduino. Інструкції, записані в секції "void loop()", виконуються відразу після інструкцій в секції "void setup()" нескінченну кількість разів, поки плата не буде вимкнена (або скинута). Тобто, вміст "void loop()" виконується від першої інструкції до останньої, потім повторно виконується від першої інструкції до останньої, потім виконується від першої інструкції до останньої, і так далі, і так далі.

```
#include <DHT>
#include <UIPEthernet>
#define DHTP 2
#define DTTYPE DHT11
int ld1 = 3;
int ld2 = 4;
int ld3 = 8;
int ld4 = 9;
int relay = 3;
int rainSensePin= 00;
DHTP dhtp(DHTP, DTTYPE);
EthernetServer svr = EthernetServer(80);
```

Бібліотека Ethernet. Дозволяє підключити Arduino Ethernet Shield або плату Arduino Ethernet (або аналогічну) до мережі Ethernet (TCP/IP). Її можна налаштувати так, щоб плата працювала як сервер (тобто постійно і безперервно прослуховувала і приймала запити від інших пристроїв в мережі, які запитують якусь послугу або дані, пропоновані нею) або як клієнт (тобто

саме плата час від часу запитує ці послуги або дані від іншого мережевого пристрою). Ця бібліотека підтримує до чотирьох одночасних з'єднань (вхідних - тобто в "режимі сервера", вихідних - тобто в "режимі клієнта" - або їх комбінацію).

```
void setup() {
  uint8_t mac[6] = {0x01,0x06,0x03,0x02,0x07,0x08};
  IPAddress myIP(138,232,213,242);
  Ethernet.begin(macaddr,IP);
  server.begin();
  dhtp.begin();
  pinMode(ld1, OUTPUT);
  pinMode(ld2, OUTPUT);
  pinMode(ld3, OUTPUT);
  pinMode(ld4, OUTPUT);
  pinMode(rel, OUTPUT);
  pinMode(SensePin, INPUT);
}
```

Вкладки інструкцій, що містяться в секціях "void setup()" і "void loop()" програми, зовсім не обов'язкові для успішної компіляції. Вони просто є способом написання коду впорядковано, зрозуміло і зручно для програміста, полегшуючи йому читання вже написаного коду і підтримуючи певну структуру при його написанні.

```
void loop() {
  String cmd;
  EthernetClient Eth_client = server.available();
    if (Eth_client){
      while (Eth_client.connected()) {
         if (Eth_client.available()) {
           char command = Eth_client.readstring();
           commandString += String(cmd);}
            delay(10);
           informations(cmdString);
             if (!Eth_client.connected()) {
               Eth_client.stop();}
         }
      }
      delay(1);
      Eth_client.stop(); }
```

Функція inform() може бути використана для надсилання шаблону, який не кодується за замовчуванням у бібліотеці. В цій функції можемо використовувати для імітації будь-якого протоколу, навіть якщо він не закодований за замовчуванням у бібліотеці. Це можна зробити, вибравши одну з команд. Таким чином, ми зможемо побачити у вікні точні значення слів, які будуть включені до масиву.

```
void inform (String cmds) {
  if(cmds.equals("TEMP")){
    server.print(dh.readTemperature());
    server.print(dh.readHumidity());
    return;
  }
  if(cmds.equals("LED1ON")){
    digitWrite(ld1, HIGH);
    retun;
  }
  if(cmds.equals("LED1_OFF")){
    digitWrite(ld1, LOW);
    retun;
  }
  if(cmds.equals("WATER")){
    int waterSenseReading = analogRead(rainSensePin);
    server.print(waterSenseRead);
    retun;
  }
  if(cmds.equals("RELAY_ON")){
    digitWrite(relay, HIGH);
    retun;
  }
  if(cmds.equals("RELAY_OFF")){
    digitWrite(relay, LOW);
    retun;
  }
}
```

Важливою особливістю Ардуїно є простота програмування, оскільки він використовує своєрідну спрощену мову С, де багато функцій було усунуто, створивши дуже просту, але дуже універсальну мову завдяки можливості додавання бібліотек відповідно до різних проектів, до яких ви хочете приступити. Зараз розглянемо програмування мережевих сокетів. Лістинг для клієнтського сокету наведено нижче.

```
public class SocketClient {
   public void clientGetData(String cmd) {
      UserData userDt = new UserData();
      try (Socket s = new Socket(169.274.213.212,81);
           BufReader in = new BufReader(new InputStreamReader(s.getInputStream()));
           DataOutputStream out = new DataOutputStream(s.getOutputStream())) {
          out.writeBytes(cmd);
          if (id.equals("TEMP")) {
userData.setTemperature(Double.parseDouble(in.readLine()));
        userData.setHumidity(Double.parseDouble(in.readLine()));
          } else if (id.equals("WATER")) {
userData.setRainSense(Integer.parseInt(in.readLine()));
          }
      } catch (IOException e) {
          clientGetData(id);
      }
   }
```

Тепер розглянемо під’єднання до БД. Нижче приведений код для побудови під’єднання до бази даних.

```
public class DBConn {
   private String url;
   private String user;
   private String password;
   private String driver;

   public DBConn() {
      driver = "com.mysql.jdbc.Driver";
      url = "jdbc:mysql://localhost:3306/registrdb";
      user = "root";
      password = "root";
   }

   public Connection getConn(){
      Class.forName(driver).newInstance();
      return DriverManager.getConnection(url, user, password);
   }
}
```

Нижче розташований програмний код класу WaterControl для роботи сигналізації води.

```
class WaterControl extends CreatingStreams {
   @Override
   public void run() {
      List<Button> butt = new ArrayList<>();
      butt.add(new AddController().getButton());
      while (!stop) {
```

```
SocketClient socketClient = new SocketClient();
UserData userData = new UserData();
socketClient.clientGetData("WATER");
if (userData.getRainSense() <= 300) {
    for (Button button : btn) {
        button.setStyle("-fx-background-color: red;");
        new UserData().setIdRoom(new Requests().getIdRoom(button.getId()));}
    new UserData().setIndicator (true);
    new Requests().insertDataSignaling("Затоплення!");
    stop();}
try {
    Thread.sleep(1000);
} catch (InterruptException e) {
    e.printStackTrace();}
```

Такий самий підхід використовується для програмування роботи інших датчиків.

3.4 Побудова інтерфейсу користувача системи безпечної експлуатації приміщень

Процес проєктування інтерфейсу користувача зазвичай вимагає поєднання ручного та автоматизованого проєктування. При автоматизованому проєктуванні вони кодуються в програму, яка запускається кожного разу, коли тестується система, що розробляється. Зазвичай це швидше, ніж ручне проєктування інтерфейсу користувача, особливо якщо воно включає регресійне проєктування інтерфейсу користувача, тобто проєктування інтерфейсу, яке передбачає повторний запуск попередніх тестів, щоб перевірити, чи зміни в програмі не призвели до появи нових помилок. Використання автоматизованого проєктування інтерфейсу користувача значно зросло за останні кілька років. Однак, проєктування інтерфейсу користувача ніколи не може бути повністю автоматизованим, оскільки такий вид проєктування інтерфейсу користувача лише перевіряє, чи робить програма те, що вона повинна робити. Практично неможливо використовувати автоматизовані тести для проєктування інтерфейсу користувача систем, які залежать від зовнішнього вигляду, або для перевірки того, що програма не має небажаних побічних ефектів.

JavaFX - це програмна платформа для створення та доставки десктопних додатків, а також Rich Internet Applications (RIA), які можуть працювати на різних пристроях. JavaFX призначена для заміни Swing в якості стандартної бібліотеки графічного інтерфейсу для Java SE. Вона дозволяє розробникам проектувати, створювати, тестувати, налагоджувати та розгортати багатофункціональні клієнтські програми.

Зовнішній вигляд JavaFX-додатків можна налаштовувати за допомогою каскадних таблиць стилів (CSS) (див. JavaFX: CSS) і (F) XML-файли можна використовувати для структурування об'єктів, що полегшує створення або розробку програми (див. FXML і контролери). Scene Builder - це візуальний редактор, який дозволяє створювати fxml-файли для інтерфейсу користувача без написання коду.

JavaFX API доступні як повністю інтегрована функція Java SE Runtime Environment (JRE) і Java Development Kit (JDK). Оскільки JDK доступний для всіх основних настільних платформ (Windows, Mac OSX і Linux), JavaFX-додатки, скомпільовані для JDK 7 і пізніших версій, також працюють на всіх основних настільних платформах. Підтримка платформ ARM також стала доступною з JavaFX 8. JDK для ARM включає основні компоненти JavaFX, графіку та елементи керування.

```
public class JFXApp extends Application {
public static void main(String [] args) {
launch( args );
}
public void init(){
// Ініціація застосунку
. . .
}
@Overide
public void start(Stage primStg) {
// Параметри сцени
. . .
primaryStage.setScene(_scene);
primaryStage.setVisible( true );
}
public void stop(){
// зупинка застосунку
. . .}}
```

У JavaFX вміст сцени представляється у вигляді ієрархічного графа вузлів сцени. У цьому прикладі кореневим вузлом є об'єкт StackPane, який є вузлом компонування зі змінним розміром. Це означає, що розмір кореневого вузла залежить від розміру сцени і змінюється, коли користувач змінює розмір сцени.

На рис. 3.17 показаний інтерфейс користувача на базі технології JavaFX для реалізації операції вибору необхідної локалізації.

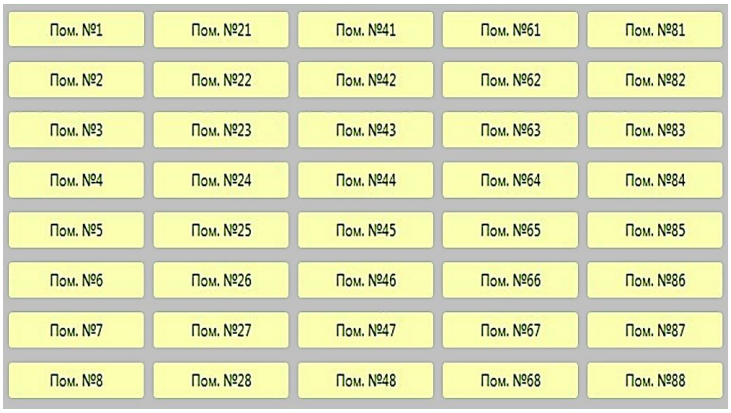

Рисунок 3.17 – Інтерфейс користувача для вибору необхідної локалізації

Створення форми є поширеним видом діяльності при розробці додатків. Цей урок навчить вас основам роботи з екранними менеджерами макетів, як додавати елементи керування на панель макетів і як створювати події введення. Будемо використовувати JavaFX для створення форми для входу в систему, показаної нижче.

```
public void initMainF() {
        int coun = 1;
        GridPane gridP = new GridPane();
        gridP.getStyleClass().add("grid-pane");
        for (int i = 0; i < 20; i++) {
            for (int j = 0; j < 30; j++) {
                Button btn = new Button();
                btn.setId(String.valueOf(coun));
                btn.setText("Пом. №" + coun++);
                btn.getStyleClass().add("button-all");
                btn.setOnAction(event -> new AdditionalController().buttonPressed(event));
                gridP.add(btn, i, j);}
                    }
}
```

Для форми входу, яку ми збираємося створити, ми будемо використовувати макет GridPane, тому що він дозволяє створити гнучку сітку з рядків і стовпців, в якій можна розмістити елементи управління. Ви можете розмістити елементи управління в будь-якій комірці сітки, і ви можете зробити так, щоб елементи управління охоплювали комірки, як вам потрібно. Додамо код перед рядком primaryStage.show().

На рис. 3.18 приведені елементи основної форми приміщення, яке було обране користувачем.

Рисунок 3.18 – Інтерфейс головної форми JavaFX для обраного приміщення

Зовнішній вигляд JavaFX-додатків можна налаштовувати. Каскадні таблиці стилів (CSS) відокремлюють зовнішній вигляд від логіки програми, щоб розробники могли зосередитися на коді. Графічні дизайнери можуть легко налаштовувати зовнішній вигляд програми за допомогою CSS. Якщо у вас є фоновий веб-дизайн або ви хочете розділити інтерфейс користувача (UI) і серверну логіку, ви можете розробити презентаційні аспекти UI на мові сценаріїв FXML, а для логіки програми використовувати код Java. Якщо ви віддаєте перевагу розробці інтерфейсу користувача без написання коду, використовуйте JavaFX Scene Builder. При проектуванні інтерфейсу користувача за допомогою javaFX Scene Builder створюється код розмітки

FXML, який можна перенести в інтегроване середовище розробки (IDE), щоб розробники могли додати бізнес-логіку.

```
<?xml version="1.0" encoding="UTF-8"?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<AnchorPane id="AnchorPane" prefHeight="220" prefWidth="220"
xmlns:fx="http://javafx.com/fxml/1"
      fx:controller="com.ex.FXMLDocContr">
   <children>
      <Button layoutX="12" layoutY="90" text="Click" onAction="#handleBtnAct" fx:id="button" />
      <Label layoutX="12" layoutY="120" minHeight="16" minWidth="69" fx:id="label" />
</children>
</AnchorPane>
```

JavaFX API доступні як повністю інтегрована функція середовища виконання Java SE (JRE) та Java Development Kit (JDK). Оскільки JDK доступний для всіх основних настільних платформ (Windows, Mac OS X і Linux), JavaFX-додатки, скомпільовані в JDK і згодом запущені на всіх основних настільних платформах, також працюють на них. Підтримка платформ ARM також стала загальнодоступною з JavaFX 8. JDK для ARM включає основні графічні та керуючі компоненти JavaFX.

Крос-платформна сумісність забезпечує узгоджене виконання для розробників і користувачів додатків JavaFX. Oracle забезпечує синхронізацію випусків і оновлень на всіх платформах і пропонує комплексну програму підтримки для підприємств, що використовують критично важливі додатки на JavaFX.

ВИСНОВКИ

При виконанні кваліфікаційної роботи спроєктоване і розроблено апаратно-програмний комплекс автоматизації безпечної експлуатації приміщень особливої важливості. Це дає можливість користувачеві контролювати параметри, які характеризують як загальний стан житлового будинка або промислової будівлі, так і окремі ділянки приміщень.

При реалізації кваліфікаційної роботи було:

- проведено загальний аналіз базових принципів функціонування мікропроцесорних комплексів;
- проведено аналіз програмного забезпечення мікропроцесору ARDUINO;
- проаналізовані аналоги систем контролю безпечної експлуатації приміщень особливої важливості;
- розроблена структурна модель автоматизованого комплексу;
- розроблені математичні моделі процесів у системі моніторингу;
- розроблена структурна схема автоматизованої системи безпечної експлуатації приміщень;
- розроблена структура бази даних системи безпечної експлуатації приміщень;
- розроблені блок-схеми алгоритмів модулів системи автоматизації безпечної експлуатації приміщень;
- розроблені процедури збірки підсистем комплексу датчиків системи безпечної експлуатації приміщень особливої важливості;
- розроблені програмні модулі системи автоматизованого контролю безпечної експлуатації приміщень особливої важливості;
- розроблено інтерфейс користувача програмної системи.

Розроблена апаратно-програмна система автоматизації безпечної експлуатації приміщень особливої важливості може бути застосована при проєктуванні спеціалізованих приміщень з особовими умовами експлуатації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Денисюк В.О., Цирульник С.М. Мікропроцесорні системи управління: навч. посіб./ В.О.Денисюк, С.М.Цирульник; Вінн. нац. аграр. ун-т. Вінниця: ТВОРИ, 2021. 204 с.
2. Скородєлов В. В. Цифрові пристрої та мікропроцесори. Архітектура та програмування мікроконтролерів: навчальний посібник / В. В. Скородєлов, О. М. Рисований, О. Ф. Даниленко, М. В. Ліпчанський. – Харків : ХВУ, 2004. – 318 с.
3. Теорія автоматичного керування : конспект лекцій з курсу / П. П. Говоров та ін. – Харків : ХНУМГ, 2012. – 221 с.
4. Офіційний сайт Arduino [Електронний ресурс]. - Режим доступу: https://www.arduino.cc.
5. Євстаф’єв А.В. Мікроконтролери AVR сімейств Tiny і Mega фірми ATMEL, 5 вид., Стер. - Видавничий дім «Додека XXI», 2013. - 560 с.
6. J. M. Hughes. Arduino: A Technical Reference. O’Reilly. 2016.
7. Arduino Швидкий старт. Перші кроки з освоєння Arduino, Видавництво: М.: Макском. 2015. - 80 с.
8. Blum J. Exploring Arduino: Tools and Techniques for Engineering Wizardry / J. Blum. – England: John Wiley & Sons, Inc., 2019. – 512 p.
9. Брайан В. Еванс, Arduino блокнот програміста, 2012. – 40 с.
10. Dimosthenis E. Bolanakis. Embedded Programming with Arduino. 2021.
11. Simon Monk. Programming Arduino: Getting Started with Sketches. McGrawHill. 2011.
12. Jonathan Oxer and Hugh Blemings. Practical Arduino. Apress. 2009.
13. Emily Gertz and Patrick Di Justo. Environmental Monitoring with Arduino. Maker Media. 2012.
14. Сопер М.Е. Практичні поради та рішення по створенню «розумного будинку». 2012. – 432с.
15. F.Perea Arduino Essentials, 2015 – 206 pages

16.Черничій М. Ю. Велика енциклопедія електрика / Черничій М.Ю. - Ескмо, 2011. - 272 с.

17.Grady Koch. Programming the Arduino. 2020.

18.Соммервілл І. Інженерія програмного забезпечення. - Видавництво Вільямс, 2013. - 624 с.

19. Дейтел Х. М., Дейтел П. Дж., Сантрі С. І. Технології програмування на Java 2. Книга 2. Розподілені додатки - ТОВ «Біном-Пресс», 2014. - 464 с.

20. Савченко І. Структурні схеми та алгоритми: Навчальний посібник. – К.: Наука, 2013. – 286 с.

21.Болл Стюарт Р. Аналогові інтерфейси мікроконтролерів. - Додека-XXI, 2015. - 360 с.

22.Д. В. Сукачов. Інфрачервоні датчики руху і присутності - реальний спосіб економії електроенергії, ТОВ «Марбел», 2015 - 141 с.

23. Гуревич В. І. Високовольтні пристрої автоматики на герконах. - Хайфа, 2011. - 368 с.

24.Гуревич В. І. Електричні реле. Пристрій, принцип дії і застосування. Настільна книга інженера. - Солон-прес, 2011. - 700 с.

25. Теребнев В.В., Артем’єв Н.С., Корольченко Д.А., Підгрушний А.В., Фомін В.І., Грачов В.А. Промислові будівлі і споруди. Серія «Протипожежний захист та гасіння пожеж». Книга 2. - Пожнаука, 2012. с. 289.

26.Г.Ф. Шанаєв, А.В. Леус. Системи захисту периметра, Security Focus, 2015.- 280с.

27.Плужников В. О. «Вплив мотивації праці на підвищення ефективності виробництва» // Економіка АПК. – 2011. - №4. – С. 137-140.

28.Таранавська Н. П., Пушкар Р. М. Менеджмент: теорія та практика. – Тернопіль: Крат-бланш, 2012. – 366 с.

ДОДАТКИ

Додаток А.

```
package com.mysoft.controller;
import com.mysoft.db.Requests;
import com.mysoft.main.Main;
import com.mysoft.pojo.UserData;
import com.mysoft.socket.SocketClient;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;
public class AdditionalController extends Requests implements
Initializable {
    @FXML
    public VBox vBoxRight;
    @FXML
    public BorderPane borderPane;
    @FXML
    public GridPane gridPane;
    @FXML
    public GridPane gridPaneClimate;
    @FXML
    public GridPane gridPaneLoad;
```

```
    @FXML
    public GridPane gridPaneSignaling;
    @FXML
    private TableView<UserData> tableView;
    @FXML
    private TableColumn<?, ?> column1;
    @FXML
    private TableColumn<?, ?> column2;
    private UserData userData;
    private static Stage stage;
    private static Button button;
    public AdditionalController() {
        userData = new UserData();
    }
    public void buttonPressed(ActionEvent event) {
        try {
            SocketClient socketClient = new SocketClient();
            Button button = (Button) event.getSource();
            setButton(button);
            switch (button.getId()) {
                case "1":
                    getStatusIndicators();
                    socketClient.clientGetData("TEMP");
                    userData.setLedOn1("LED1_ON");
                    userData.setLedOff1("LED1_OFF");
                    userData.setLedOn2("LED2_ON");
                    userData.setLedOff2("LED2_OFF");
                    userData.setLedOn3("LED3_ON");
                    userData.setLedOff3("LED3_OFF");
                    userData.setLedOn4("LED4_ON");
                    userData.setLedOff4("LED4_OFF");
                    userData.setLoadOn1("RELAY_ON");
                    userData.setLoadOff1("RELAY_OFF");
Main().initAdditionalForm(button.getText(),
userData.getTemperature(), userData.getHumidity());
                    onCloseOperation(getStage());
```

```
                    break;
                case "2":
                    getStatusIndicators();
                    userData.setLedOn1("9");
                    userData.setLedOff1("9");
                    userData.setLedOn2("9");
                    userData.setLedOff2("9");
                    userData.setLedOn3("9");
                    userData.setLedOff3("9");
                    userData.setLedOn4("9");
                    userData.setLedOff4("9");
                    new
Main().initAdditionalForm(button.getText(), 23, 50);
                    onCloseOperation(getStage());
                    break;
                default:
                    getStatusIndicators();
                    userData.setLedOn1("9");
                    userData.setLedOff1("9");
                    userData.setLedOn2("9");
                    userData.setLedOff2("9");
                    userData.setLedOn3("9");
                    userData.setLedOff3("9");
                    userData.setLedOn4("9");
                    userData.setLedOff4("9");
                    new
Main().initAdditionalForm(button.getText(), 0, 0);
                    onCloseOperation(getStage());
                    break;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    private Stage getStage() {
        return stage;
```

```
    }
    public static void setStage(Stage stage) {
        AdditionalController.stage = stage;
    }
    @Override
    public void initialize(URL location, ResourceBundle
resources) {
        column1.setCellValueFactory(new
PropertyValueFactory<>("date"));
        column2.setCellValueFactory(new
PropertyValueFactory<>("situation"));
        loadDataBaseData(button.getId());
        tableView.setItems(getData());
    }
    @FXML
    private void handleButtonUpdateTable(){
        loadDataBaseData(button.getId());
    }
    @FXML
    private void handleButtonDeleteAll() {
        deleteAllDataInRoom();
        loadDataBaseData(button.getId());
    }
    public Button getButton() {
        return button;
    }
    private void setButton(Button button) {
        AdditionalController.button = button;
    }
}

package com.mysoft.controller;
import com.mysoft.main.Main;
import javafx.fxml.FXML;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
```

```
import javafx.scene.text.Text;
import java.io.IOException;
public class LoginController {
    @FXML
    private Text errorReport;
    @FXML
    public TextField textLogin;
    @FXML
    private PasswordField textPassword;
    private Main main;
    public LoginController() {
        main = new Main();
    }
    @FXML
    private void handleOpenRegistrationForm() throws IOException
{
        main.initRegistrationForm();
        textLogin.clear();
        textPassword.clear();
        errorReport.setText(null);
    }
    @FXML
    private void handleGoSignIn() throws IOException {
        if (new
SignInController().isLoginValid(textLogin.getText(),
textPassword.getText())) {
            main.initMainForm();
        } else {
            errorReport.setText("Ошибка! Неверный Логин или
Пароль!");
            textLogin.clear();
            textPassword.clear();
        }
    }
}
package com.mysoft.controller;
```

```
import com.mysoft.db.RegistrationDBConnection;
import javafx.fxml.FXML;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class RegistrationController {
    @FXML
    private TextField textLogin;
    @FXML
    private TextField textPassword;
    private Stage dialogStage;

    public void setDialogStage(Stage dialogStage) {
        this.dialogStage = dialogStage;
    }
    @FXML
    private void handleAddNewUsers() {
        String login = textLogin.getText();
        String password = textPassword.getText();
        try (Connection connection = new
RegistrationDBConnection().getConnected();
             PreparedStatement preparedStatement =
connection.prepareStatement("INSERT INTO
registration.registration(login, password) VALUES (?,?)")) {
            preparedStatement.setString(1, login);
            preparedStatement.setString(2, password);
            preparedStatement.executeUpdate();
            textLogin.clear();
            textPassword.clear();
            dialogStage.close();
        } catch (IllegalAccessException | InstantiationException
| SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        }
```

```
    }
}


package com.mysoft.controller;
import com.mysoft.db.RegistrationDBConnection;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
class SignInController {
    boolean isLoginValid(String login, String password) {
        try (Connection connection = new
RegistrationDBConnection().getConnected();
             PreparedStatement preparedStatement =
connection.prepareStatement("SELECT * FROM
registration.registration WHERE login=? AND password=?")) {
            preparedStatement.setString(1, login);
            preparedStatement.setString(2, password);
            ResultSet resultSet =
preparedStatement.executeQuery();
            return resultSet.next();
        } catch (IllegalAccessException | InstantiationException
| SQLException | ClassNotFoundException e) {
            e.printStackTrace();
            return false;
        }
    }
}


package com.mysoft.db;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class RegistrationDBConnection {
```

```
    private final String driver;
    private final String url;
    private final String user;
    private final String password;
    public RegistrationDBConnection() {
        driver = "com.mysql.jdbc.Driver";
        url = "jdbc:mysql://localhost:3306/registration";
        user = "root";
        password = "root";
    }
    public Connection getConnected() throws
ClassNotFoundException, InstantiationException,
IllegalAccessException, SQLException {
        Class.forName(driver).newInstance();
        return DriverManager.getConnection(url, user, password);
    }
}


package com.mysoft.db;
import com.mysoft.controller.AdditionalController;
import com.mysoft.pojo.DateTime;
import com.mysoft.pojo.UserData;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.stage.Stage;
import java.sql.*;
public class Requests extends DateTime {
    private String query;
    private UserData userData = new UserData();
    private ObservableList<UserData> data =
FXCollections.observableArrayList();
    // Получение статусов индикаторов
    protected void getStatusIndicators() {
```

```
        query = "SELECT information.id_room,room.room,
information.led1, information.led2, information.led3,
information.led4, " +
                "information.cond, information.hea,
information.pool, information.vent, " +
                "information.load1, information.load2,
information.load3, information.load4, " +
                "information.fire, information.water,
information.smoke, information.sign, information.zam " +
                "FROM room, information " +
                "WHERE room.id_room = information.id_room " +
                "AND room.room = ?";
        try (Connection connection = new
RoomInformationDBConnection().getConnected();
             PreparedStatement preparedStatement =
connection.prepareStatement(query)) {
            preparedStatement.setString(1, "Помещение №" + new
AdditionalController().getButton().getId());
            ResultSet resultSet =
preparedStatement.executeQuery();
            while (resultSet.next()) {
userData.setIdRoom(resultSet.getInt("id_room"));
userData.setStatusLed1(resultSet.getBoolean("led1"));
userData.setStatusLed2(resultSet.getBoolean("led2"));
userData.setStatusLed3(resultSet.getBoolean("led3"));
userData.setStatusLed4(resultSet.getBoolean("led4"));
userData.setStatusCond(resultSet.getBoolean("cond"));
userData.setStatusHea(resultSet.getBoolean("hea"));
userData.setStatusPool(resultSet.getBoolean("pool"));
userData.setStatusVent(resultSet.getBoolean("vent"));
userData.setStatusLoad1(resultSet.getBoolean("load1"));
userData.setStatusLoad2(resultSet.getBoolean("load2"));
userData.setStatusLoad3(resultSet.getBoolean("load3"));
userData.setStatusLoad4(resultSet.getBoolean("load4"));
userData.setBtnStatusFire(resultSet.getBoolean("fire"));
userData.setStatusWater(resultSet.getBoolean("water"));
```

```
userData.setBtnStatusSmoke(resultSet.getBoolean("smoke"));
userData.setBtnStatusSign(resultSet.getBoolean("sign"));
userData.setBtnStatusZam(resultSet.getBoolean("zam"));
        } catch (IllegalAccessException | InstantiationException
| SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
    // При закрытии дополнительной формы, в БД заносятся статусы
индикаторов
    protected void onCloseOperation(Stage stage) {
        query = "UPDATE `room`.`information` " +
                "SET `led1`= ?, `led2`= ?, `led3`= ?, `led4`= ?,
"
                "`cond`= ?, `hea`= ?, `pool`= ?, `vent`= ?, " +
                "`load1` = ?, `load2` = ?, `load3` = ?, `load4`
= ?," +
                "`fire` = ?, `water` = ?, `smoke` = ?, `sign` =
?, `zam` = ? " +
                "WHERE `id_room`=?;";
        stage.setOnCloseRequest(event -> {
            try (Connection con = new
RoomInformationDBConnection().getConnected();
                 PreparedStatement preparedStatement =
con.prepareStatement(query, Statement.RETURN_GENERATED_KEYS)) {
                preparedStatement.setBoolean(1,
userData.isStatusLed1());
                preparedStatement.setBoolean(2,
userData.isStatusLed2());
                preparedStatement.setBoolean(3,
userData.isStatusLed3());
                preparedStatement.setBoolean(4,
userData.isStatusLed4());
                preparedStatement.setBoolean(5,
userData.isStatusCond());
```

```
                preparedStatement.setBoolean(6,
userData.isStatusHea());
                preparedStatement.setBoolean(7,
userData.isStatusPool());
                preparedStatement.setBoolean(8,
userData.isStatusVent());
                preparedStatement.setBoolean(9,
userData.isStatusLoad1());
                preparedStatement.setBoolean(10,
userData.isStatusLoad2());
                preparedStatement.setBoolean(11,
userData.isStatusLoad3());
                preparedStatement.setBoolean(12,
userData.isStatusLoad4());
                preparedStatement.setBoolean(13,
userData.isBtnStatusFire());
                preparedStatement.setBoolean(14,
userData.isStatusWater());
                preparedStatement.setBoolean(15,
userData.isBtnStatusSmoke());
                preparedStatement.setBoolean(16,
userData.isBtnStatusSign());
                preparedStatement.setBoolean(17,
userData.isBtnStatusZam());
                preparedStatement.setInt(18,
userData.getIdRoom());
                preparedStatement.executeUpdate();
            } catch (IllegalAccessException |
InstantiationException | ClassNotFoundException | SQLException
e) {
                e.printStackTrace();
            }
        });
    }
    // Загрузка базы данных для таблицы детальной информации
    protected void loadDataBaseData(String numRoom) {
```

```
        String query = "SELECT
date,situation,ind_fire,ind_smoke,ind_sign,ind_water,ind_zam " +
                "FROM room.detailed_information, room.room " +
                "WHERE detailed_information.id_room =
room.id_room " +
                "AND room.room = \"Помещение №" + numRoom +
"\";";
        try (Connection connection = new
RoomInformationDBConnection().getConnected();
             PreparedStatement preparedStatement =
connection.prepareStatement(query);
             ResultSet resultSet =
preparedStatement.executeQuery()) {
            getData().clear();
            while (resultSet.next()) {
                getData().add(new UserData(
                        resultSet.getString("date"),
                        resultSet.getString("situation")
                ));
userData.setIndicatorFire(resultSet.getBoolean("ind_fire"));
userData.setIndicatorSmoke(resultSet.getBoolean("ind_smoke"));
userData.setIndicatorSign(resultSet.getBoolean("ind_sign"));
userData.setIndicatorWater(resultSet.getBoolean("ind_water"));
userData.setIndicatorZam(resultSet.getBoolean("ind_zam"));
        } catch (IllegalAccessException | InstantiationException
| SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
    // Вставка данных сигнализации
    public void insertDataSignaling(String signaling) {
        query = "INSERT INTO detailed_information (date,
situation, ind_fire, ind_smoke, ind_water, ind_sign, ind_zam,
id_room) " +  "VALUES (?, ?, ?, ?, ?, ?, ?, ?);";
        try (Connection connection = new
RoomInformationDBConnection().getConnected();
```

```
            PreparedStatement preparedStatement =
connection.prepareStatement(query,
Statement.RETURN_GENERATED_KEYS)) {
            preparedStatement.setString(1, getDateAndTime());
            preparedStatement.setString(2, signaling);
            preparedStatement.setBoolean(3,
userData.isIndicatorFire());
            preparedStatement.setBoolean(4,
userData.isIndicatorSmoke());
            preparedStatement.setBoolean(5,
userData.isIndicatorWater());
            preparedStatement.setBoolean(6,
userData.isIndicatorSign());
            preparedStatement.setBoolean(7,
userData.isIndicatorZam());
            preparedStatement.setInt(8, userData.getIdRoom());
            preparedStatement.executeUpdate();
        } catch (IllegalAccessException | InstantiationException
| SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
    // Получение ИД комнаты
    public int getIdRoom(String nameRoom) {
        int id = 0;
        query = "SELECT id_room FROM room WHERE room =
\"Помещение №" + nameRoom + "\"";
        try (Connection connection = new
RoomInformationDBConnection().getConnected();
             PreparedStatement preparedStatement =
connection.prepareStatement(query);
             ResultSet resultSet =
preparedStatement.executeQuery()) {
            while (resultSet.next()) {
                id = resultSet.getInt("id_room");
            }
```

```
        } catch (IllegalAccessException | InstantiationException
| SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        }
        return id;
    }
    // Удаление всех данных из таблицы в выбранной комнате
    protected void deleteAllDataInRoom() {
        query = "DELETE FROM detailed_information WHERE id_room
= ?";
        try (Connection connection = new
RoomInformationDBConnection().getConnected();
             PreparedStatement preparedStatement =
connection.prepareStatement(query)) {
            preparedStatement.setInt(1, userData.getIdRoom());
            preparedStatement.executeUpdate();
        } catch (IllegalAccessException | InstantiationException
| SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
    protected ObservableList<UserData> getData() {
        return data;
    }
}
```