Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

# Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: **«Комп'ютерна модель аналізу індексів акцій на фондовому ринку»**

Захищено на засіданні
Атестаційної комісії № 38
протокол № __ від __.12.2021 р.
Оцінка _____ / _____
Голова Атестаційної комісії
д.т.н., професор
_____ Мінухін Сергій Володимирович

**Виконав:**
Галузь знань: 12 – Інформаційні технології
за спеціальністю 123 – Комп'ютерна інженерія.

Анжуров Валентин Євгенович _____

**Керівник:**
д.т.н., с.н.с., професор кафедри теоретичної та прикладної системотехніки
Толстолузька О.Г. _____

**Рецензент:**
д.т.н., доцент, завідувач кафедри безпеки інформаційних систем і технологій
Рассомахін С.Г. _____

Харків – 2021

# АНОТАЦІЯ

Кваліфікаційна робота складається зі вступу, трьох розділів, висновку, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає 63 сторінки, з яких 46 сторінок основної частини що містять 22 рисунки, 23 найменування списку використаних джерел на 3 сторінках і 4 додатки на 10 сторінках.

Об'єкт дослідження: процес кластеризації даних компаній на фінансовій біржі методом k-середніх

Предмет дослідження: комп'ютерні моделі та методи препроцесінгу в Data Mining

Мета: підвищення ефективності аналізу індексів акцій на фондовому ринку за рахунок використання комп'ютерної моделі кластеризації даних компаній на фінансовій біржі методом k-середніх

Методи дослідження: процес кластеризації даних методом k-середніх

Результати дослідження: комп'ютерна модель в вигляді програмного продукту, що має простий інтерфейс та оптимальний час роботи

В ході даної роботи були розглянуті концепції Data Mining, що стосуються препроцесінгу, а саме його типу – кластеризації. Були розглянуті та досліджені деякі моменти написання моделі препроцесінгу Під час написання моделі були застосовані сучасні технології. В роботі наведені порівняння можливих шляхів побудови даної моделі та дослідження стосовного кожного з них.

**Ключові слова**: Data Mining, пепроцесінг, кластеризація, метод k-середніх, комп'ютерна модель, Python.

# ABSTRACT

The qualifying work consists of an introduction, three sections, a conclusion, a list of sources used and four appendices. The total volume of the work is 63 pages, of which 46 pages of the main part contain 22 figures, 23 names of the list of used sources on 3 pages and 4 appendices on 10 pages.

Object of research: the process of clustering these companies on the financial exchange by the method of k-means

Subject of research: computer models and methods of preprocessing in Data Mining

Objective: to increase the efficiency of the analysis of stock indexes in the stock market through the use of a computer model of clustering of data of companies on the financial exchange by the method of k-means

Research methods: the process of clustering data by k-means

The results of the study: a computer model in the form of a software product that has a simple interface and optimal operating time

In the course of this work, the concepts of Data Mining related to preprocessing, namely its type - clustering, were considered. Some aspects of writing a preprocessing model were considered and investigated. Modern technologies were used during the writing of the model. The paper compares possible ways of constructing this model and studies relevant to each of them.

**Keywords**: Data Mining, reprocessing, clustering, k-means method, computer model, Python.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application programming interface – Програмний інтерфейс аплікації

ОС – операційна система

БД – база даних

PCA – Principal component analysis – Аналіз головних компонентів

HTTP – Hypertext transfer protocol – Протокол передачі гіпертексту

HTTPS - Hypertext transfer protocol secure – Захищений протокол передачі гіпертексту

IDEF0 – Integrated Definition for Function Modeling – Інтегроване визначення для моделювання функцій

# ВСТУП

Кожна людина шукає шляхи зберегти/збільшити її зароблений тим чи іншим чином фінансовий капітал. Сьогодні тема фондового ринку та його цінних паперів має актуальний характер, тому що це є один з найстаріших засобів інвестування. Фондовий ринок як явище з'явився в Америці в 1970 році, що є роком виникнення першої фондової біржі. Компанії зрозуміли, що капітал можна надбати не лише шляхом продажу власної продукції, а також шляхом вкладів звичайних людей в бізнес компаній. Таким чином, ці люди ставали співвласниками мізерної частини бізнесу та мали відповідний заробіток з цього.

Питання інвестицій вільного капіталу сьогодні хвилює велику кількість людей в багатьох частинах світу. Найцікавіші з точки зору звичайного інвестора перспективи інвестування:

- Збереження та зрощення капіталу, що існує
- Захист від інфляції
- Вихід на ранню пенсію шляхом забезпечення власних потреб за рахунок бонусів, що виплачують компанії своїм вкладникам
- Додатковий прибуток до заробітку від основної діяльності

Враховуючі ці аспекти, можна стверджувати, що інвестиції та дослідження, де інвестиції є основною моделлю, є дуже актуальними в сьогодення.

З причини того, що інформація про акції фондового ринку занадто велика та різноманітна, потрібно вміти швидко і зручно орієнтуватися в ній. Для цього, безумовно, підходить така галузь, як Data Science і, зокрема, її підгалузь Data Preprocessing. Розглядаючи Data Preprocessing, можна виділити такий метод, як кластеризація даних, де вхідні дані групуються в кластери (групи) не за класифікаційною ознакою, а за ознакою належності до певної групи.

Для досягнення успіху в інвестуванні, потрібно мати змогу виявляти більш перспективні для інвестування компанії, де можна було б отримати найбільший прибуток. Саме цю проблему і вирішує комп'ютерна модель, що використовує метод кластеризації k-середніх для групування кластеру, що містить найбільш перспективні компанії. Також, система може проводити кластеризаційний аналіз попередніх років, так як в реальному світі інвестицій потрібно мати змогу аналізувати минулі ситуації на фондовому ринку, щоб розуміти, як діяти сьогодні.

Комп'ютерні моделі мають потужність спроможну відтворювати складні обчислення. Все це марно, коли людина, що використовує цю систему для написання програми, не вміє користуватися цією потужністю. Велика кількість систем працює невдало та повільно саме через це, результатом чого є нестабільний продукт, з яким неможливо працювати. Тому необхідно вміти використовувати ресурси, що видаються машиною для забезпечення швидкої роботи комп'ютерної моделі. Задача розробника полягає в написанні ефективного коду, що оптимально використовує ресурси системи.

На щастя, сучасні методи препроцесінгу та кластеризації працюють таким чином, що їх достатньо впевнено можна використовувати у високонавантажених системах. Тим не менш, потрібно вивчати та досліджувати нові методи роботи з ними задля досягнення більш ефективних результатів у майбутньому.

Робота включає у себе дослідження можливого використання кластеризації та методу k-середніх для аналізу минулого та поточного станів фондового ринку з метою рекомендації покупки акцій з найбільшою вірогідністю росту у майбутньому

**Мета дослідження** – підвищення ефективності аналізу індексів акцій на фондовому ринку за рахунок використання комп'ютерної моделі кластеризації даних компаній на фінансовій біржі методом k-середніх

**Завдання дослідження:**

- дослідити методи препроцесінгу в Data Mining;
- дослідити методи кластеризації в Data Mining;
- розробити комп'ютерну модель аналізу;
- скласти інструкцію для користування.

**Об'єктом дослідження** є процес кластеризації даних компаній на фінансовій біржі методом k-середніх

**Предметом дослідження** є комп'ютерні моделі та методи препроцесінгу в Data Mining

**В роботі виконані наступні завдання:**

1. Проаналізовано існуючі методи препроцесінгу даних у Data Mining.
2. Досліджені формати вхідних та вихідних даних, інтерфейсу користувача.
3. Розроблено комп'ютерну модель аналізу даних фондового ринку
4. Розроблено інструкцію для користування.

# РОЗДІЛ 1.
# DATA MINING ТА КЛАСТЕРИЗАЦІЯ

Дані в комп'ютерних моделях збираються в різних форматах та станах. Можливі втрачені дані, можливі некоректні дані або ж взагалі «шум». Усі подальші етапи обробки інформації залежать від того, наскільки якісно був проведений препроцесінг зібраної інформації.

## 1.1 Data Mining

Data mining (з англ. видобуток даних) - це процес напівавтоматичного аналізу великих баз даних з метою пошуку корисних фактів. Зазвичай, поділяють на задачі класифікації, моделювання та прогнозування [**Error! Reference source not found.**]. На сучасних підприємствах, в дослідницьких проектах або в інтернеті утворюються великі обсяги даних. Глибинний аналіз даних здійснюється автоматично шляхом застосування методів математичної статистики, штучних нейронних мереж, теорії нечітких множин або генетичних алгоритмів. Метою аналізу є виявлення правил та закономірностей, наприклад, статистичних подій. Так, наприклад, можливо виявити зміни у поведінці клієнтів або груп клієнтів для покращення стратегії підприємства.

До Data Mining відносять наступні поняття:

1. Безпосередня обробка великого обсягу даних.
2. Здатність алгоритму до виявлення патерів патернів у даних. При обробці утворюються групи даних, що схожі за відтінками або іншими ознаками.
3. Процесінг може робити прогнози щодо фінального результату. Під час обробки алгоритм може на певному етапі зробити прогноз результату останнього етапу, орієнтуючись на вже оброблену інформацію всіх етапах.

4. Знаходження потрібної інформації. Збір даних алгоритмами виявляє ознаки даних і з цього користувач отримує найбільш релевантну інформацію щодо свого запиту.

Не зважаючи на те, що збір статистики та процес Data Mining виглядають однаковими, можна побачити різні властивості між ними:

1. Data Mining є незалежним процесом від стороннього втручання людини та не вимагає її присутності взагалі.
2. Неможливо порівнювати об'єм даних, що опрацьовує статистичний метод та Data Mining в силу того, що у порівнянні в Data Mining, об'єм даних для метода дуже малий.

Data Mining процес ділиться на 4 етапи:

1. По-перше, визначається основна проблема, яку належить розв'язати процесу.
2. Відбувається збір на попередня підготовка даних для подальшого аналізу. Необхідно обрати найбільш ефективний спосіб збору об'єму даних. Далі зібрані дані передаються до етапу попередньої обробки, де відсікається шум, заповнюються втрачені частини, відбувається фінальна компресія.
3. Будується модель та проводиться оцінка даних. Модель має бути використана для машинної обробки кінцевих даних та отримання певного результату.
4. Фінальні дані мають бути використані для вирішення раніше поставленої проблеми. На їх основі будується висновок щодо роботи моделі в цілому, а також на цьому етапі стає зрозуміло чи вдалося вирішити поставлену задачу.

Необхідно зазначити те, що Data Mining є лише методологічними рекомендаціями щодо збору та обробки великої кількості даних, а не готовим

програмним рішенням. Людина, що використовує Data Mining, має чітко оперувати правилами та поняттями методології, щоб отримати найбільш сприятливий результат та вирішити задачу.

### 1.2 Препроцесінг

Препроцесінг даних – метод збирання інформації, що використовується для конвертування швидко зібраних даних у правильний формат, що може бути ефективно використаним у наступних етапах процесу Data Mining. Препроцесінг складається з наступних кроків [**Error! Reference source not found.**]:

1) Спочатку дані підлягають очищенню. Інформація може мати невідповідності та відсутні частини. Для обробки цієї інформації потрібно замінити шум, втрачені дані, тощо:
    a. Шумні дані – дані, які не відносяться до безпосереднього предмету дослідження або ж не несуть ніякого змістовного навантаження;
    b. Втрачені дані – дані, що не були зібрані з тієї або іншою причини. Як правило заповнюються у відповідності до сусідніх даних.
2) Перетворення інформації. Цей етап полягає в перетворенні очищених даних у потрібний для дослідника формат. Етап передбачає наступні способи:
    a. Дискретизація – заміна необроблених ділянок даних числовими атрибутами;
    b. Покоління ієрархії концепції – атрибути мають бути перетворені за поточного рівня на рівень віще в ієрархії;
    c. Нормалізація – робиться для масштабування значення даних у визначеному діапазоні;

d. Вибірковий метод атрибутів – стратегія будування нових атрибутів із заданого набору;

e. Кластеризація – процедура будування групи даних за схожими ознаками.

3) Компресія даних [**Error! Reference source not found.**]. Зібрані дані, як правило, представлені дуже великим об'ємом, що безумовно робить процес дослідження більш трудомістким, та не завжди ці часові витрати виправдовуються результатом. Спосіб компресії передбачає зменшення об'єму даних із зберіганням цінних властивостей для дослідника. Нижче наведені деякі способи компресії даних:

a. Зменшення чисельності – передбачає зберігання моделі даних замість цілісних даних;

b. Вибирання атрибутної підмножини. Використовувати потрібно лише актуальні дані. Для проведенні вибору атрибутів потрібно використати рівень значущості та p-значення атрибутів. Атрибут, що має p-значення більше за рівень значущості, може бути відкинутим із зібраного об'єму даних;

c. Кубічна агрегація інформації – застосовується до інформації для побудування куба даних;

d. Зменшення розмірності – проводиться шляхом зменшення за допомогою механізмів кодування. Ці механізми можуть бути втратними або безвтратним і, якщо після реконструкції зі зжатої інформації можуть бути отримані вхідні дані, тоді таке зменшення називають безвтратним. Інакше воно вважається втратним. Два найбільш популярні методи зменшення розмірності – перетворення вейвлетів та PCA (аналіз основного компоненту).

### 1.3 Кластеризація

Не можна недооцінити значення кластеризація при роботі с препроцесінгом даних в Data Mining. Утворення груп (кластерів) інформація за схожими ознаками широко застосовується в багатьох галузях, де потрібна обробка великої кількості даних [**Error! Reference source not found.**].

Кластеризація – процес утворення груп з певного набору об'єктів, де групування базується на порівнянні за ознаками, характеристиками тощо. В Data Mining методології розглядаються дані, які реалізують перший алгоритм поєднання, що найбільш підходить до аналізу поточних даних.

Кластеризація має особливість, що об'єкт не має бути належним до того чи іншого кластеру, викликаючи цей тип жорсткого угруповання секціонування. Тим не менш, м'який поділ вказує на те, що кожен об'єкт належним чином належить кластеру. Більш детальний розгляд вказує на те, що можуть бути створені об'єкти з кількома кластерами, щоб змусити об'єкт брати участь лише в одному кластері, і навіть будувати ієрархічні дерева на основі відносин між групами.

Існують наступні доводи того, що кластеризація є ефективним етапом препроцесінгу:

1) Кластеризація має функціонал для роботи з різними типами вхідних об'єктів. Алгоритми мають змогу обробляти будь-який тип даних. Наприклад, можуть бути застосовані до числового діапазону, двійкових даних або категоріальних даних.

2) Ідентифікація груп за формою атрибутів. Процес кластеризації повинен бути універсальним, коли мова іде про довільну ідентифікацію кластеру. Цей процес обмежений лише вимірюванням відстані від значення атрибуту до центроїда кластера. Дана відстань має тенденцію бути виявленою невеликим сферичним кластером [**Error! Reference source not found.**].

3) Кластеризація може обробляти неочищені від шуму на попередніх етапах дані. Бази с інформацією можуть мати галасливі або хибні дані. Інші алгоритми можуть бути чутливими до таких даних. Результатом їх роботи є низькоякісні кластери.

4) Масштабованість. Для роботи з великим обсягом інформації необхідні високо масштабні методи кластерного аналізу.

5) Висока розмірність. Методи кластеризації можуть обробляти як інформацію за малою розмірністю, так дані з високим простором розмірності.

## 1.4 Метод k-середніх

### 1.4.1 Алгоритм

Метод k-середніх – ітеративний алгоритм, що переслідує мету розділення об'єму вхідних даних на окремі кластери, кожен з яких об'єднує об'єкті, схожі за характеристиками. Метод намагається розбити дані на максимально великі за кількістю об'єктів групи шляхом розрахунку квадратів відстаней між значеннями об'єктів та значеннями центроїдів кластерів. Відстань дорівнює середньому арифметичному всіх точок даних, що належить кластеру. Однорідність даних в кластері можна досягнути шляхом збирання даних для цього кластера, які мають однорідні характеристики [**Error! Reference source not found.**].

Алгоритм метода k-середніх описаний нижче:

1) Визначається кількість кластерів.

2) Обираються випадкові значення для кожного кластера.

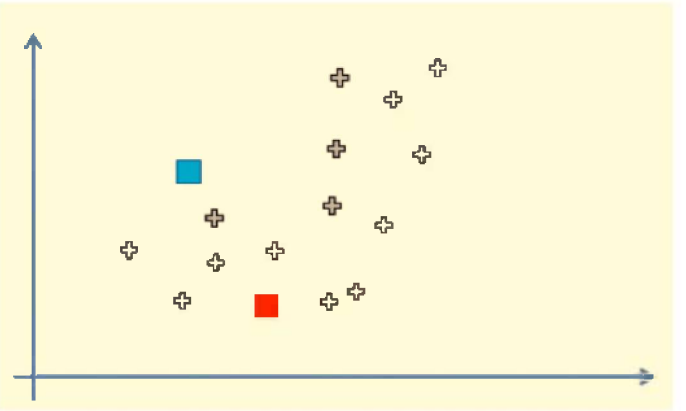


Рисунок 1.1 – випадкові центроїди

3) Кожне значення вхідних даних співвідноситься з центроїдом, до якого квадратична відстань є найменшою.

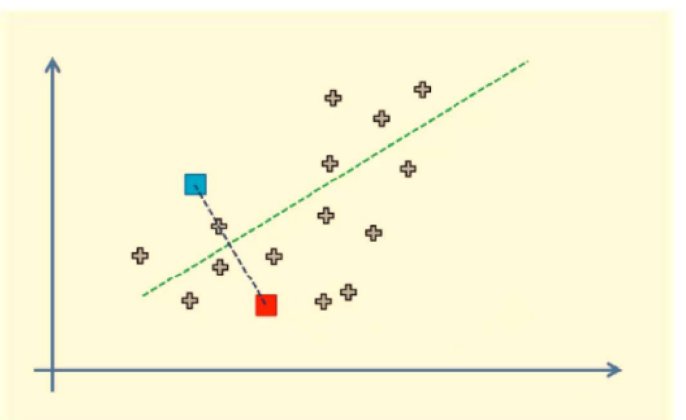


Рисунок 1.2 – формування початкових кластерів

4) Значення центроїдів перераховуються.

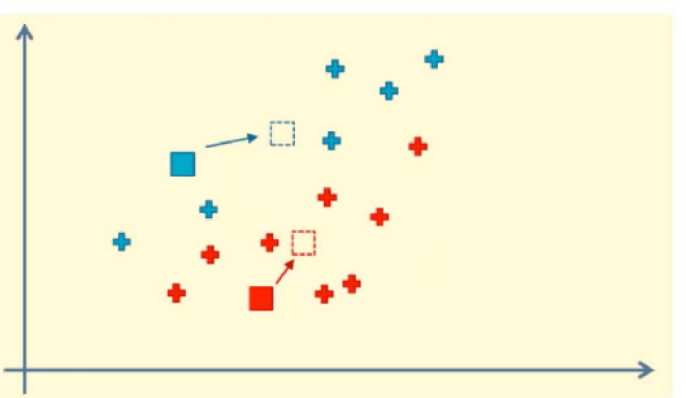


Рисунок 1.3 – перераховані центроїди

5) Відбувається перевизначення кластера для кожної точки з урахуванням нових значень кластерів.

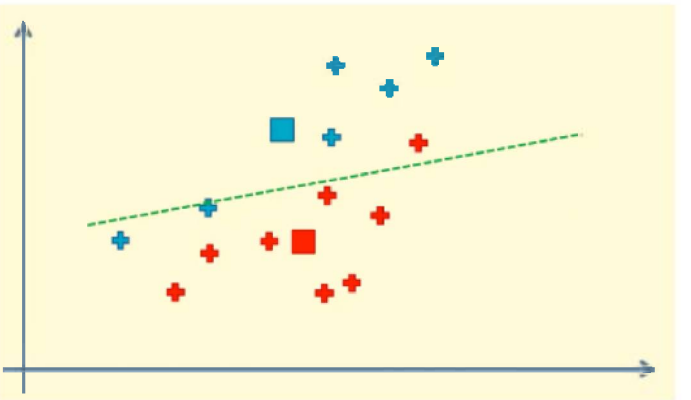


Рисунок 1.4 – перерозподілені точки

6) Кроки 4-5 повторюються, доки перераховані значення центроїдів не зійдуться із попередніми.

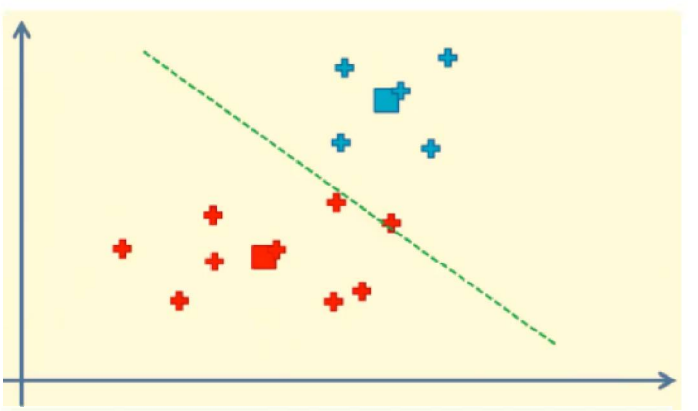


Рисунок 1.5 – результат роботи методу

### 1.4.2 Переваги та недоліки методу

Взявши в роботу будь-який метод або алгоритм кластеризації, можна зрозуміти що він має переваги та недоліки у порівнянні з іншими. Потрібно розуміти, що кожна проблема має певні характеристики, і кожен з методів може бути більш менш ефективним у різних обставинах. Щоб вирішити проблему цієї роботи, було визначено, що метод k-середніх є рішенням, що відповідає поточним обставинам [**Error! Reference source not found.**].

Переваги методу:

1. Спроможність роботи з великим об’ємом інформації.
2. Простий у реалізації.
3. Гарантує конвергенцію – близьке розташування даних за схожими ознаками чи характеристиками.
4. Має здатність узагальнюватись до скупчень різного розміру та форм, таких як еліптичні скупчення. Як веде себе метод, коли кластери мають різний розмір та щільність, зображено на Рис. 1.6. З лівого боку знаходяться інтуїтивні кластери, з правого ті, що є результатом методу k-середніх. Це є доказом того факту, що метод не є орієнтовним на різноманітність форм, а на співвідношення до центроїдів.

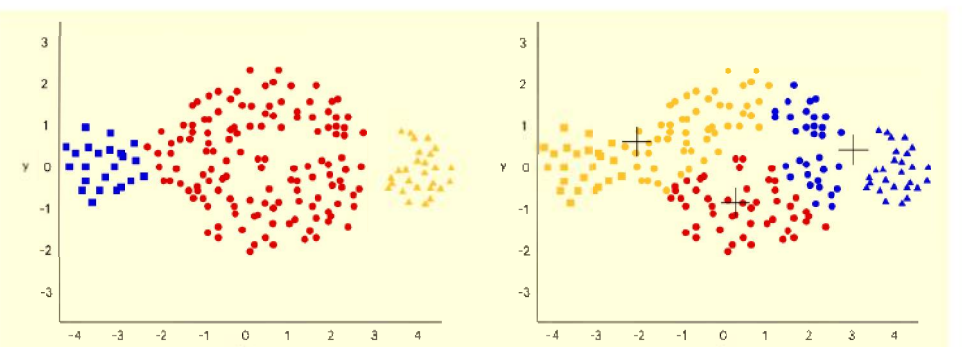


Рисунок 1.6 – Інтуїтивні кластери та кластери із методу k-середніх

5. Легко адаптивний до нових видів вхідних даних. Робота метода не залежить від того якої складності дані він оброблює. Основна задача – сформувати чіткі кластери.
6. Алгоритм піддається модифікації. Для проведення кластеризації кластерів, таких як на Рис. 1.6, можна адаптувати метод k-середніх. На Рис. 1.7 чорна лінія зображує межі кластеру після роботи методу. З лівого боку рисунка

зображена відсутня узагальненість. Це призводить до неінтуїтивної межі кластера. На центральній частині зображене отримання інтуїтивного кластеру різної величини. На правій – результатом є еліптичний кластер

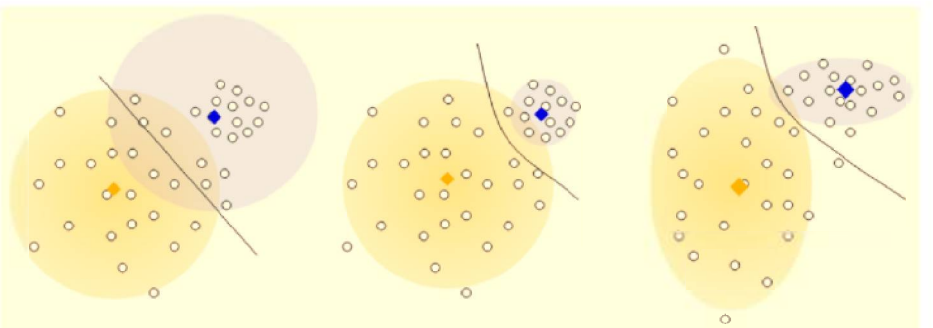


Рисунок 1.7 – Результат адаптації методу

Недоліки методу:

1. Від користувача вимагається кількість початкових центроїдів для початку роботи. Це вимагає присутності людини.
2. Алгоритм має певні проблеми, коли фінальні кластери матимуть різну величину та щільність.
3. Величини центроїдів можуть бути змінені невірно за рахунок шумних даних, або ж шум може взагалі мати власний кластер, якщо він не був видалений раніше. Отже, попередня обробка дуже важлива для коректної роботи методу k-середніх.
4. Чутливість до початкових значень центроїдів. Коли мова йде про маленьку кількість центроїдів (до 4-х), можна зменшити цю залежність, виконуючи метод декілька разів, вказавши різні початкові значення. Зі збільшенням кількості початкових центроїдів потрібно модифікувати метод таким чином, щоб похибка в результатах була мінімальною.

5. Коли обсяг даних збільшується, міра подібності на основі квадратичної відстані переходить у постійну константу між будь-якими вхідними даними. Необхідним в даній ситуації є зменшення розміру шляхом аналізу даних для характеристик, або шляхом спектральної кластеризації. Відношення стандартного відхилення до середньої квадратичної відстані між вхідними даними стрімко зменшується зі збільшенням кількості початкових кластерів. Це зменшення значить те, що метод став менш ефективним [**Error! Reference source not found.**].

## 1.5 Вибір технологій

Для того, щоб розпочати процес розробки комп'ютерної моделі, потрібно визначити технології для реалізації. Далі потрібно обрати найбільш ефективні та зручні засоби серед існуючих, проаналізувати актуальність технології на предмет використання її іншими розробниками. Далі наведені технології, що використані в процесі розробки даної системи.

### 1.5.1 Python

Існує багато мов програмування, що мають можливість інтеграції з процесом Data Mining. Далі наведений список найпоширеніші з цих мов програмування:

- Java;
- C++;
- Python;
- C#;
- JavaScript;
- PHP;

- Ruby;
- Perl;
- Go;
- Swift;
- Groovy.

Після проведеного аналізу було прийняте рішення обрати Python – інтерактивну та об'єктно-орієнтовану мову програмування. Python є розробкою простого для навчання засобу програмування. Цього вдалося досягнути шляхом того, що код, написаний на цій мові, дуже схожий на просту людську мову англійською. Гвідо ван Россум почав розробляти Python наприкінці 80-х років працівником Голландського Інституту математики та інформатики. Через широкий досвід розробників, Python набув багато функціоналу, схожого з мовами Algol, C, C++, Modula-3, SmallTalk.

Починаючі з 2010 року Python набуває більшої популярності у зв'язку с тим, що є універсальним засобом для будь-якої потреби (Див Рис. 1.8)

Завдяки популярності, за допомогою Python часто можна вирішити будь-які проблеми, які можуть виникнути в процесі розробки продукту будь-якої складності. Спільнота Python сильна, і її члени продовжують постійно розробляти покращення та налагодження технічної сторони. Python також має багато корпоративних спонсорів, які успішно популяризують мову та її методи розробки. Серед них такі технологічні гіганти як Google, що також використовую мову при розробці своїх комп'ютерних та наукових моделей [**Error! Reference source not found.**].

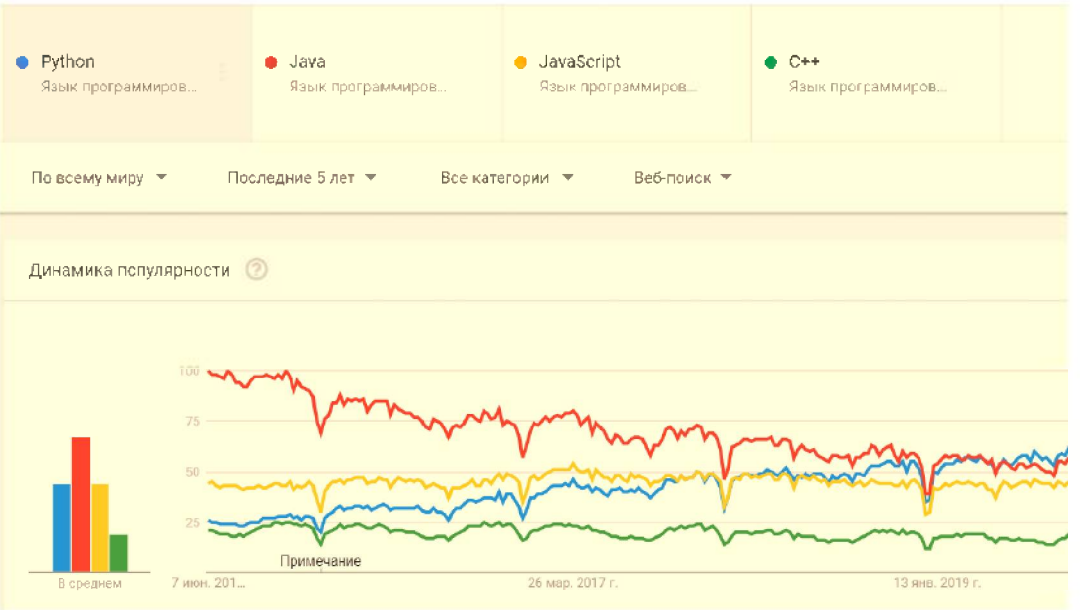



Рисунок 1.8 – Вибірка Google.com

Python був розроблений таким чином, щоб його було легко зрозуміти. Це спрощує написання нового коду Python та прискорює розробку програмного забезпечення Python. Це означає, що менше часу витрачається на боротьбу з проблемами налагодження програм і більше часу створення продукту безпосередньо.

Одна з найвідоміших особливостей Python полягає в тому, що час виконання коду є відносно повільним порівняно з іншими мовами. Однак є рішення цієї проблеми. Коли час виконання програми має високий пріоритет, Python дозволяє інтегрувати інші, більш ефективні мови у ваш код. Це оптимізує вашу швидкість, не змушуючи вас переписувати всю кодову базу з нуля. Слід зазначити, що найцінніший ресурс - це час виконання програми, а час, який розробники витрачають на розробку.

Ще однією перевагою Python є широкий вибір бібліотек та фреймворків, які він пропонує для різних ситуацій розробки. Їхнє використання підвищує ефективність роботи команди розробників, оскільки часто потрібні речі не

потрібно писати з нуля. Мова має засоби для багатьох процесів програмного забезпечення:

- Machine Learning;
- Data Science;
- Візуалізація даних;
- Складний аналіз даних.

Те ж саме стосується фреймворків, що прискорюють розробку проектів та програм [**Error! Reference source not found.**].

Python є інтуїтивно зрозумілим, тому що виглядає як проста англійська мова. Це спрощує розробку та підтримку програмного забезпечення. Крім того, Python має чіткий синтаксис і не вимагає такої кількості рядків коду, як Java або, наприклад, C для досягнення порівнянних результатів. Скорочення часу, що витрачається на перевірку коду, є безцінним, тому що збережений час можна витратити на розробку.

Python - найпопулярніша мова програмування, яку використовують мільйони інженерів для Data Science продуктів. Об'єктно-орієнтована мова програмування широко використовується в організації великих обсягів складних даних. Маючи динамічну семантику і непомірні можливості, Python також широко використовується для написання сценаріїв обробки великих обсягів даних різної складності.

Масштабованість кодової бази непередбачувана. Скільки реальних користувачів ви можете охопити, не можна сказати на початку проекту, тому важливо оцінити, наскільки серйозною має бути база коду. Тому Python є оптимальним вибором через його надійність і масштабованість. Деякі з найбільших продуктів, наприклад Microsoft, обрали Python саме з цієї причини.

### 1.5.2 Pandas

Pandas - це програмна бібліотека, написана для мови програмування Python для маніпуляції та аналізу даних. Зокрема, він пропонує структури даних та операції для маніпулювання числовими таблицями та часовими рядами. Це безкоштовне програмне забезпечення, випущене під ліцензією BSD з трьох пунктів. Назва походить від терміна «панельні дані», економетричного терміна для наборів даних, які включають спостереження протягом кількох періодів часу для одних і тих самих людей. Його назва — це гра на фразі «аналіз даних Python». Вес Маккінні почав створювати те, що стане бібліотекою в AQR Capital, коли працював там дослідником з 2007 по 2010 рік [**Error! Reference source not found.**].

Альтернативи Pandas використовують той самий функціонал та користуються досить великою популярністю серед ком'юніті розробників. Такими альтернативами є Pandasql, NumPy, Scipy, IPy, Matplotlib.

Тим не менш, Pandas має переваги серед наступного функціоналу:

- Об'єкт DataFrame для маніпулювання даними з інтегрованим індексуванням
- Інструменти для читання та запису даних між структурами даних у пам'яті та різними форматами файлів
- Вирівнювання даних та інтегрована обробка відсутніх даних
- Зміна та поворот наборів даних
- Нарізка на основі міток, химерна індексація та піднабір великих наборів даних
- Вставка та видалення стовпців структури даних
- Групування за механізмом, що дозволяє виконувати операції поділу-застосування-об'єднання над наборами даних

- Об'єднання наборів даних
- Ієрархічна вісь індексування для роботи з даними великого розміру в структурі даних меншого розміру
- Функціональність часових рядів: генерація діапазону дат і перетворення частоти, статистика рухомого вікна, лінійна регресія рухомого вікна, зсув і відставання дати.
- Забезпечує фільтрацію даних.

Pandas дозволяє імпортувати дані з різних форматів файлів, таких як значення, розділені комами, JSON, SQL та Microsoft Excel. Бібліотека дозволяє виконувати різноманітні операції маніпулювання даними, такі як об'єднання, зміна форми, виділення, а також очищення даних і функції сперечання даних.

### 1.5.3 Microsoft Excel

Microsoft Excel – це електронна таблиця, розроблена Microsoft для Windows, macOS, Android та iOS. Він містить обчислення, інструменти створення графіків, зведені таблиці та мову програмування макросів під назвою Visual Basic для додатків (VBA). Це була дуже широко застосовувана електронна таблиця для цих платформ, особливо з версії 5 в 1993 році, і вона замінила Lotus 1-2-3 як промисловий стандарт для електронних таблиць. Excel є частиною пакету програмного забезпечення Microsoft Office [**Error! Reference source not found.**].

Microsoft Excel має основні функції всіх електронних таблиць у процесі Data Mining, використовуючи сітку комірок, упорядковану в пронумеровані рядки та стовпці з літерними іменами, щоб організувати маніпуляції з даними, як-от арифметичні операції. Він має батарею функції, які відповідають статистичним, інженерним та фінансовим потребам. Це дозволяє розділяти дані, щоб переглянути їх залежності від різних факторів для різних точок зору

(за допомогою зведених таблиць і менеджера сценаріїв). Зведена таблиця — це потужний інструмент, який може заощадити час на аналізі даних. Він робить це шляхом спрощення великих наборів даних за допомогою полів зведеної таблиці. Він має аспект програмування, Data Mining для додатків, що дозволяє користувачеві використовувати широкий спектр числових методів, наприклад, для розв'язування задач кластерного аналізу, а потім звітувати про результати в електронну таблицю. Він також має різноманітні інтерактивні функції, що дозволяють використовувати інтерфейси, які можуть повністю приховати електронну таблицю від користувача, тому електронна таблиця представляє себе як так звана програма або система підтримки прийняття рішень (DSS) через спеціально розроблений інтерфейс користувача для наприклад, аналізатор запасів або взагалі як інструмент проектування, який ставить запитання користувачам і надає відповіді та звіти.

### 1.5.4 Telegram Bot API

Боти — це сторонні програми, які працюють всередині Telegram месенджеру (найпопулярніший месенджер на території СНД). Користувачі можуть взаємодіяти з ботами, надсилаючи їм повідомлення, команди та вбудовані запити. Вони керують своїми ботами за допомогою HTTPS-запитів до нашого API бота. Використавши це, можна швидко та якісно реалізувати клієнтську частину комп'ютерної моделі. Більш того, всі, в кого є Telegram, матимуть змогу користуватися системою

Боти працюють наступним чином. Боти Telegram — це спеціальні облікові записи, для налаштування яких не потрібен додатковий номер телефону. Користувачі можуть взаємодіяти з ботами двома способами:

1) Надіславши повідомлення та команди ботам, відкривши з ними чат або додавши їх до груп.

2) Надіславши запити безпосередньо з поля пошуку. Це дозволяє надсилати вміст від вбудованих ботів безпосередньо в будь-який чат, групу чи канал.

Повідомлення, команди та запити, надіслані користувачами, передаються програмному забезпеченню, запущеному на серверах. Сервер-посередник обробляє шифрування та зв'язок із Telegram API. Спілкування з сервером відбувається через простий HTTPS-інтерфейс, який пропонує спрощену версію API Telegram.

### 1.5.5 Heroku хостинг-платформа

Комп'ютерну модель на основі TelegramBotAPI потрібно розмістити в WWW для широкого доступу для користувачів. Для цього необхідні засоби хостинг-платформи. Серед найпопулярніших: Vercel, Netlify, Heroku. Був обраний останній маючи низку переваг перед конкурентами:

- 550 годин/місяць роботи Dynos (1000 після прив'язки банківської картки)
- 512 Мб ОЗП на 1 контейнер
- CI/CD, CLI

Heroku — це хмарна платформа як сервіс (PaaS), що підтримує кілька мов програмування. Одна з перших хмарних платформ, Heroku, розроблялася з червня 2007 року, коли підтримувала лише мову програмування Ruby, але тепер підтримує Java, Node.js, Scala, Clojure, Python, PHP і Go. З цієї причини кажуть, що Heroku є платформою поліглотів, оскільки вона має можливості для розробника створювати, запускати та масштабувати програми подібним чином на більшості мов.

Програми, які запускаються на Heroku, зазвичай мають унікальний домен, який використовується для маршрутизації HTTP-запитів до правильного контейнера програми. Кожен з них розподіллено по "динаміці", яка

складається з кількох серверів. Сервер Git Heroku обробляє завантаження сховища програм від дозволених користувачів. Усі служби Heroku розміщені на платформі хмарних обчислень Amazon EC2.

### 1.5.6 Кросплатформеність

Слід зазначити, що кросплатформеність надто важлива, щоб її ігнорувати. Неможливо передбачити в якій операційній системі буде працювати модель, тому необхідно забезпечити правильну роботу моделі на всіх існуючих платформах. Всі інструменти, що використовуються в цій роботі, кросплатформні і, залежно від операційної системи, не вимагають будь-яких додаткових дій з боку користувача.

Microsoft Excel – це програма, яка працює з форматом XLSX на платформі Windows. Однак аналоги цієї програми є на кожній платформі і є можливість відкрити файл XLSX для роботи.

Python і всі його бібліотеки можуть працювати на будь-якій платформі та відображати один і той самий висновок програми.

## 1.6 Огляд системи

Задачею даної роботи є розробка комп'ютерної моделі препроцесінгу даних фондового ринку з метою аналізу ситуації для інвестора. Методом препроцесінгу даних в даній системі є метод кластеризації k-середніх. Алгоритм роботи системи описаний далі:

1) Вхідними даними програми є лише рік. Це реалізовано з метою можливості аналізу ситуації за поточний та минулі роки. Можливий діапазон років – 1970-поточний рік. Див. Рис. 1.9 та Рис 1.10

2) TelegramBotAPI отримує рік і передає його серверному коду, де далі працює програма

3) Отримавши рік, програма виконує наступні кроки:

    a. Збирає назви крупніших компаній

    b. Збирає індекси цих компаній

    c. Проводить кластеризацію отриманих значень індексів

    d. Формує вихідний файл з рекомендаціями

4) Вихідний файл складається з 3 частин, кожна з яких представлена окремим кластером – результатом процесу кластеризації. Кластери виділені відповідними кольорами для зручності користувача:

    a. Перший зелений – кластер із компаніями, що рекомендовані для інвестування

    b. Другий жовтий – кластер із компаніями, інвестування в які залишені на передбачення інвестора

    c. Третій червоний – не рекомендовані для інвестування компанії

    На Рис.1.11 та Рис.1.12 зображені межі кластерів для більшого розуміння як виглядають дані вихідного файлу

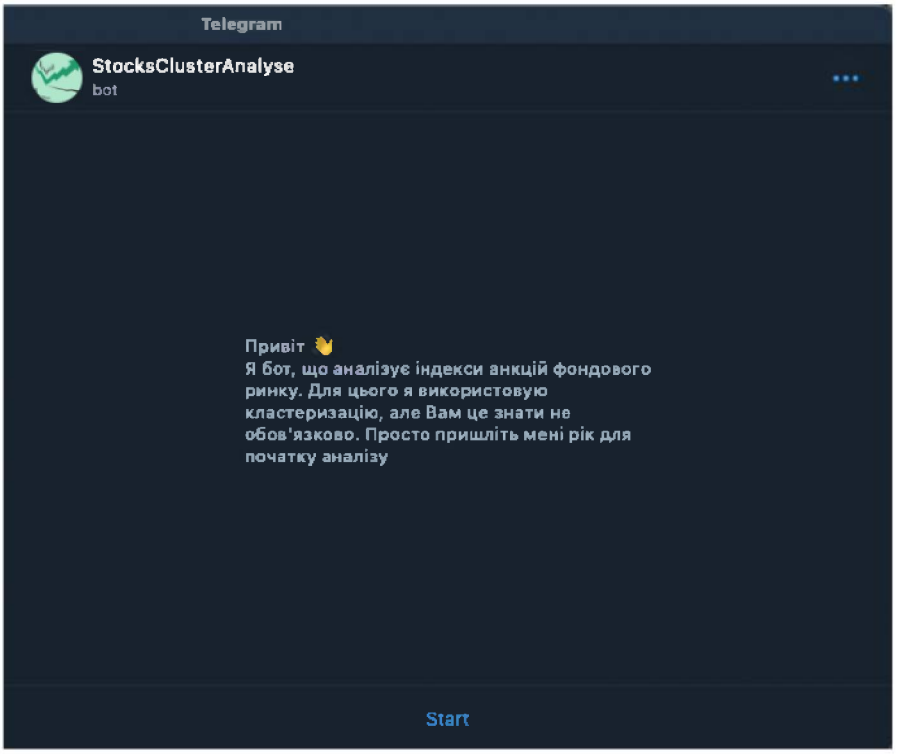



Рисунок 1.9 – Привітання боту

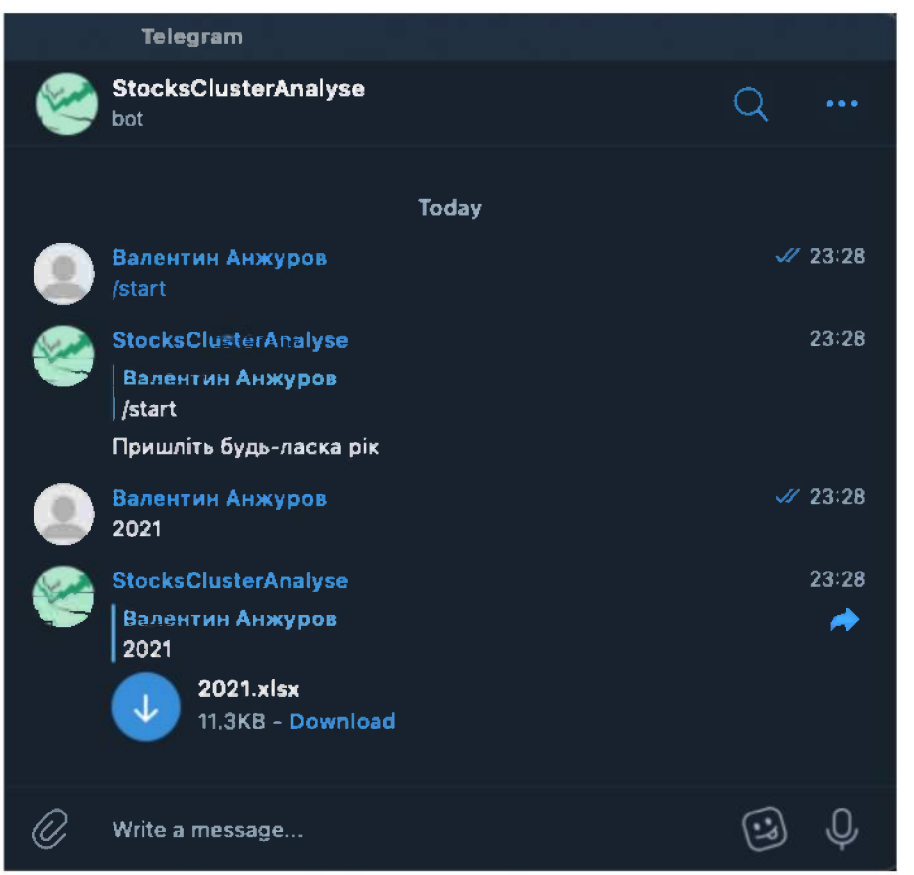



Рисунок 1.10 – Приклад інтерактивної роботи з ботом

| | |
|---|---|
| ABC | 12,8 |
| BWA | 12,87 |
| MS | 12,95 |
| NOC | 13 |
| NWL | 13,16 |
| BK | 13,39 |
| LMT | 13,42 |
| WFC | 13,45 |
| STT | 13,48 |
| GPS | 13,79 |
| CZR | 13,99 |
| TPR | 14,16 |
| BAC | 14,21 |
| SNA | 14,54 |
| ADM | 14,64 |

Рисунок 1.11 – Межа 1-го та 2-го кластерів

| | |
|---|---|
| CMS | 21,36 |
| TER | 21,36 |
| EBAY | 21,41 |
| GPC | 21,46 |
| WDC | 21,48 |
| WEC | 21,59 |
| AMGN | 21,62 |
| CSCO | 21,78 |
| KHC | 21,87 |
| ANTM | 21,9 |
| ORLY | 21,93 |
| ED | 22,03 |
| NSC | 22,17 |
| AEE | 22,2 |
| UNP | 22,3 |

Рисунок 1.12 – Межа 2-го та 3-го кластерів

**Висновки до розділу 1.**

В даному розділі були розглянуті основні концепції Data Mining. Також, був проведений огляд та аналіз методів препроцесінгу даних в Data Mining. Більш детально була звернута увага на метод кластеризаціїї k-середніх.

Були розглянуті технології, використані в процесі розробки комп'ютерної моделі даної роботи.

Більш того, був проведений огляд системи, засобів для створення комп'ютерних моделей

Розглянувши концепції Data Mining та метод k-середніх, можна зробити висновок, що цей метод є дуже зручним для вирішення задані аналізу індексів акцій на фондовому ринку

# РОЗДІЛ 2.
# РОЗРОБКА КОМП'ЮТЕРНОЇ МОДЕЛІ АНАЛІЗУ ІНДЕКСІВ АКЦІЙ НА ФОНДОВОМУ РИНКУ

Кінцевою метою продукту розробки є комп'ютерна модель аналізу індексів акцій фондового ринку, що повинна бути здатна обробити великий обсяг інформації за найкоротший час. Система має бути якомога простішою у використанні користувачем при збереженні обчислювальної здатності. Також, було б добре, щоб система складалася з найменшого обсягу програмного коду, так як менший об'єм коду спрощує майбутнє супроводження продукту.

## 2.1 Структура комп'ютерної моделі

SOLID — це абревіатура складена з перших літер п'яти базових принципів об'єктно-орієнтованого програмування та дизайну, запропонована Робертом Мартіном. Принципи SOLID використовують для дизайну та розробки таких програмних систем, які, з великою ймовірністю, зможуть тривалу годину розвиватися, розширятися та підтримуватися [**Error! Reference source not found.**].

S – Принцип єдиної відповідальності (single responsibility principle). Для кожного класу має бути визначено єдине призначення. Всі ресурси, необхідні для його здійснення, повинні бути інкапсульовані в цей клас і підпорядковані тільки цьому завданню.

O - Принцип відкритості/закритості (open-closed principle). «Програмні сутності мають бути відкриті для розширення, але закриті для модифікації».

L - Принцип підстановки Лісков (Liskov substitution principle). «об'єкти у програмі повинні бути замінені на екземпляри їх підтипів без зміни правильності виконання програми». також контрактне програмування. Похідний клас має бути взаємозамінним із батьківським класом.

I - Принцип розподілу інтерфейсу (interface segregation principle). «багато інтерфейсів, спеціально призначених для клієнтів, краще, ніж один інтерфейс загального призначення».

D – Принцип інверсії залежностей (dependency inversion principle). «Залежність на абстракціях. Немає залежності на щось конкретне»

Конструкція системи, що є метою цієї кваліфікаційної роботи, складається з різних частин. Кожна з цих частин виконую одну і тільки одну власну функцію. Незалежно від того, якої складності є система, вона повинна виконувати деякі правила вірного дизайну програмного забезпечення. Більш детально у даному підрозділі можна розглянути принцип S – єдиної відповідальності. Слідуючи цьому правилу, пропонується переслідувати «горизонтальне» розширення систем, де відбувається не розширення існуючого функціоналу, а збільшення кількості маленьких блоків програмного забезпечення. Таким чином, будь-яка проблема, що виникає під час роботи може бути швидко усунена.

На Рис. 2.1 можна розглянути частими даної комп'ютерної моделі. Кожен прямокутник визначає окремий функціональний блок, в середині якого позначена дія, яку він виконує.

- Модуль зчитування вхідних даних. Даний блок створює зчитування року роботи фондового ринку шляхом інтерактивного спілкування з користувачем через консоль месенджеру Telegram. Для цього він використовує засоби Telegram Bot API
- Валідація вхідних даних. Цей модуль здійснює перевірку вхідних параметрів для подальшої обробки системою. Рік роботи фондового ринку повинен бути цілим числом і зі значенням дійсного фондового ринку. Також потрібна перевірка того, що в даний рік біржа працювала (почала роботу з 1970 року)

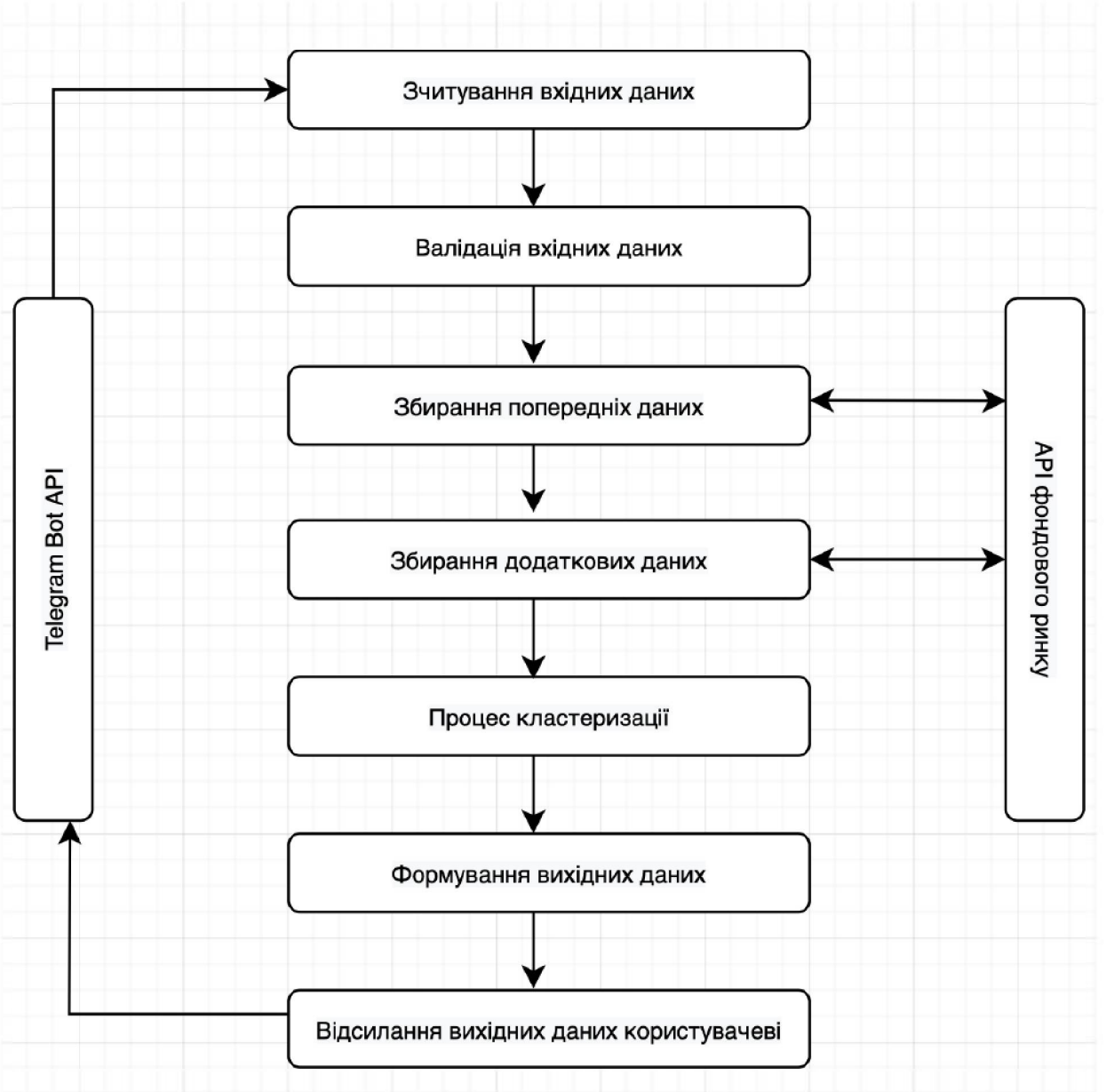



Рисунок 2.1 – Блоки системи

- Збирання попередніх даних. В цьому функціональному блоці необхідно визначити кандидатів для наступних етапів аналізу. Шляхом взаємодії з API фондового ринку система отримує велику кількість тикерів (ідентифікаторів) компаній, починаючі з найбільших і закінчуючи найменшими, так як вони можу бути перспективнішими за гігантів

- Збирання додаткових даних. Даний блок здійснює збір індексів компаній, що були зібрані попереднім блоком. Відбувається робота з іншим API фондової біржі, який повертає значення індексу P/E для кожного емітенту
- Модуль кластеризації даних. Цей функціональний блок є основним в даній системі і він містить алгоритм кластеризації k-середніх, що застосовується до попередньо зібраних значень P/E індексу. Зчитування даних алгоритмом виконується дуже швидко, основною часовою затрато є процес кластеризації. Алгоритм роботи даного модуля більш детально зображений на Рис. 2.2

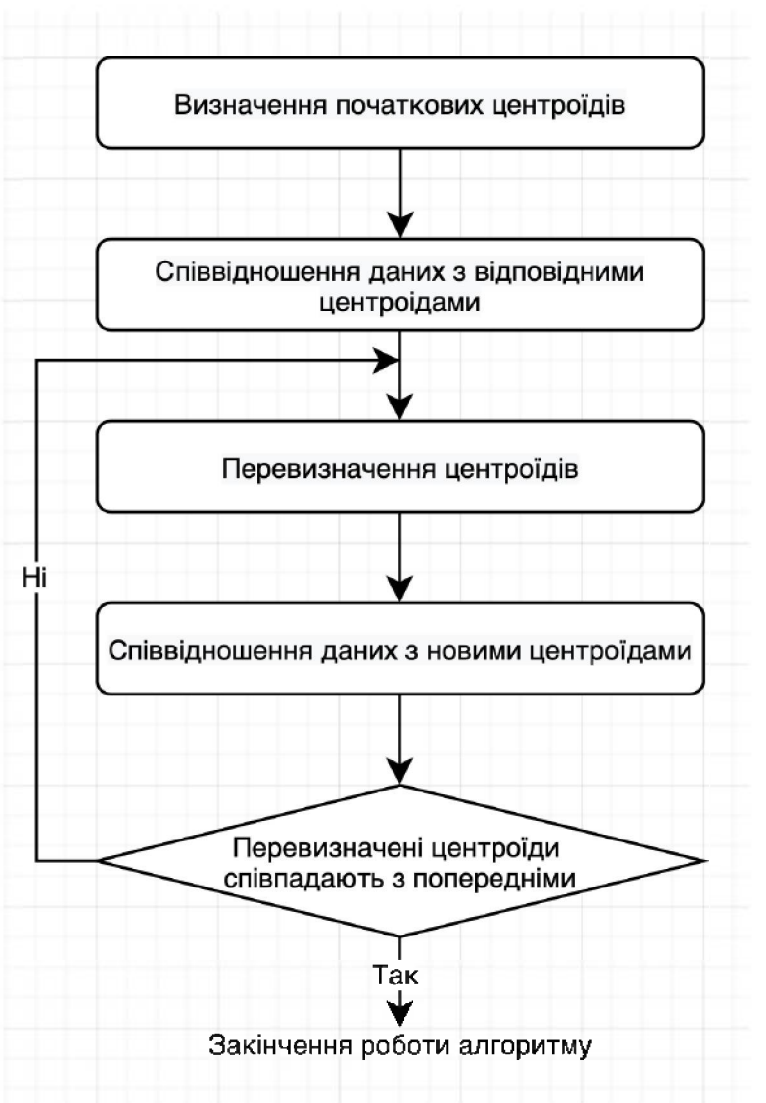



Рисунок 2.2 – Схема модуля кластеризації даних

- Формування вихідних даних. Результат кластеризації представлений Pandas конструкцією і не може бути представлений користувачеві. Даний модель конвертує конструкцію в excel файл, який легко аналізується людиною, що не має досвіду розробника
- Відсилання вихідних даних користувачеві. Цей блок, так само як і перший в даному списку, працює з користувачем через Telegram консоль засобами Telegram Bot API. Блок відсилає excel файл користувачеві шляхом HTTPS протоколу

Нижче на Рис. 2.3 наведена IDEF0 [**Error! Reference source not found.**] діаграма комп'ютерної моделі кластеризації

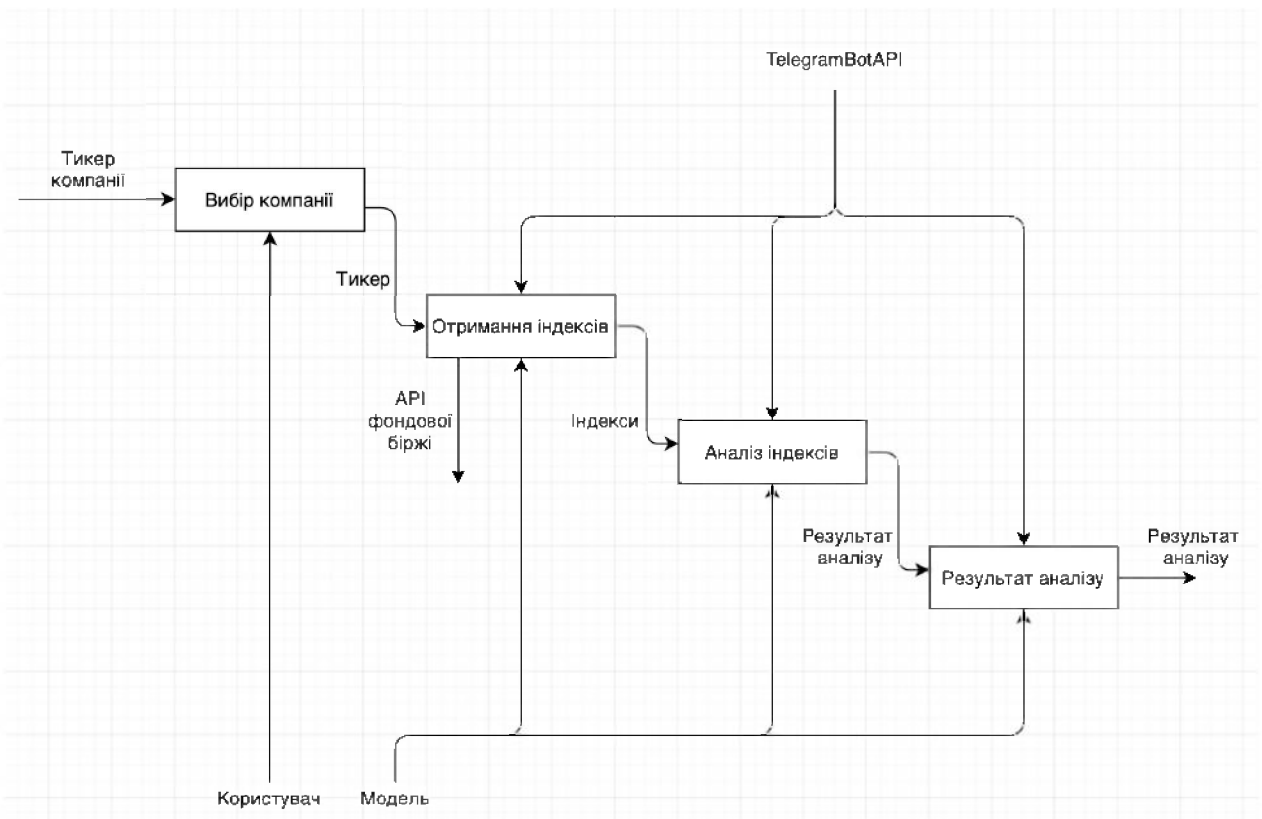



Рисунок 2.3 - модель в нотації IDEF0

Правильно структурована модель складається із частин, що можливо використовувати в інших проектах. Отже, якщо ми маємо модуль, що містить алгоритм зчитування даних користувача, ми можемо використовувати його в інших проектах зі схожою функціональністю. Це стверджує факт того, що модуль спроектований таким чином, що він не залежить від сусідніх або інших модулів програмного забезпечення. Кожен із перелічених вище модулів працює саме таким чином. Розділення системи на перелічені частини достатньо, щоб вважати архітектуру придатною для подальшого зручного розширення. Крім того, набору таких блоків є достатнім для виконання поставленої задачі аналізу даних фондового ринку. Небажано розбивати модель на докладніші блоки, оскільки це може ускладнити навігацію в середині коду.

## 2.2 Вхідні та вихідні дані

Однією з найважливіших етапів проектування комп'ютерної моделі будь-якого призначення є вибір формату вхідних та вихідних даних. Складність структури цих даних прямим чином впливає на те, чи є інтерфейс системи простим і не перенавантаженим для кінцевого користувача. Натомість, надто простий формат даних може спричинити втрату деталей результату опрацювання інформації моделлю.

Існує не так багато популярних форматів вхідних/вихідних даних. Далі проведений аналіз для вибору найоптимальніших з них з урахуванням особливостей роботи даної системи:

1. Звичайний текст. Дані представлені звичайним текстом, що по суті є послідовністю кодів переведених процесом кодування у символи. Даний варіант, як правило, потребує додаткової валідації, так як користувач не має змоги обирати з передбачених варіантів

2. Текстовий файл. Є близьким аналогом варіанту 1, за виключення того, що текст зберігається, безпосередньо, в текстовому файлі, що має своє власне кодування

3. Excel файл. Є близьким аналогом варіанту 2, тільки дані структуровані сіткою, в якій доступ до даних здійснюється методом звернення до комірки. Кожна з комірок є незалежною, або ж залежною від іншим. Даний процес залишається на розсуд користувача того чи іншого excel файлу

4. База даних. Є дуже серйозним варіантом зберігання даних будь-якої складності. Більш того, сервер бази даних є доступним будь-кому (при бажання) з будь-якої точки світу, де є Інтернет та засіб спілкування за HTTP протоколом. Дані в базі даних структуровані таблицями та колонками в цих таблицях. Також, можлива типізація даних для запобігання валідації даних при кожній операції модифікації. Для роботи з базами даних застосовується окремий синтаксис SQL, що потребує додаткового вивчення розробником

Кожен з перелічених типів вхідних/вихідних даних є популярним та визнаним спілкою розробників програмного забезпечення. Проте, кожен з них має переваги і недоліки в залежності від комп'ютерної моделі, в якій вони застосовуються. Розглянемо ці типи в контексті даного програмного забезпечення:

1. Текст є надто важким для сприйняття користувачем форматом для вихідним даних. Важко уявити наскільки складним може бути аналіз «сирого» тексту, що містить складну структуру даних в середині. Також, важко працювати з простим текстом з перспективою майбутнього розширення програмного забезпечення.

Проте, даний формат є дуже вигідним з точки зору використання його для формату вхідних даних. Дана система приймає лише рік роботи

фондового ринку. Інші варіанти для даної мети є занадто ускладненими, та зроблять роботу з системою лише важчою

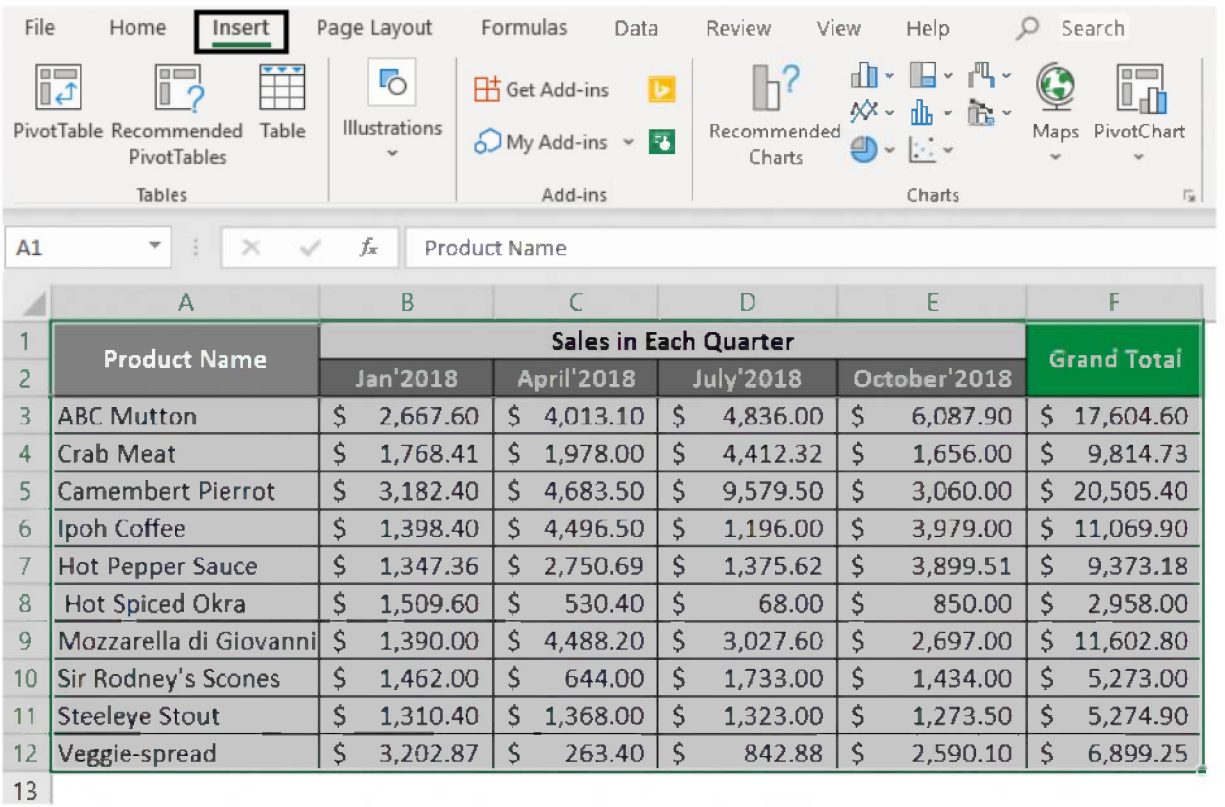
2. Файл з текстом. Так само, як і текст є дуже важким для сприйняття користувачем, враховуючі комплексність вихідних даних в даній комп'ютерній моделі. Також, звичайні текстові файли не мають засобів для виділенні тексту різними кольорами, що може дуже допомогти при аналізі вихідної інформації. Враховуючі вищесказане, можна зробити висновок, що текстовий файл не є оптимальним варіантом ні для вхідних, ні для вихідних даних

3. Excel файл. Після проведення проектування комп'ютерної моделі було прийняте рішення, що excel-файл є найзручнішим типом вихідних даних. Дані у такому файлі можуть бути розділені на аркуші, відсортовані та відфільтровані. Також, є засіб виділення даних різними кольорами. До того ж, Python має велику кількість засобів роботи з такими excel-файлами, що є достатньо оптимізованими та налагодженими. Приклад даних у excel файлі зображено на Рис. 2.3

4. База даних, як правило, використовуються для роботи за даними, що мають дуже складну структуру [**Error! Reference source not found.**][**Error! Reference source not found.**]. Також системи, де застосовані бази даних, зазвичай є високонавантаженими і основну частину спрощення роботи з багатьма потоками бере на себе безпосередньо база даних. Приклад складної структури бази даних наведено на Рис. 2.4

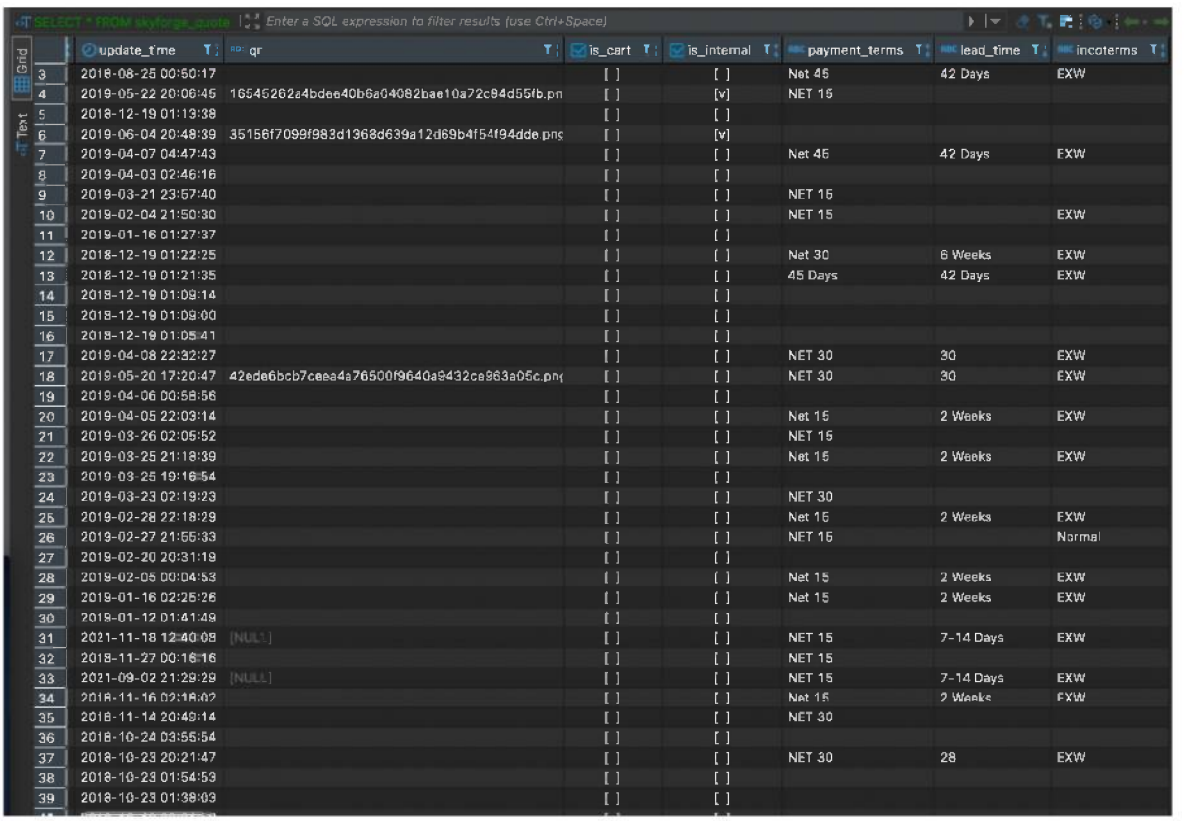
База дана потребує окремого сервера, що потребує окремої роботи, такої як розгортання серверу, його налагодження. Також, потрібно, щоб усі розробники на проекті вміли працювати з діалектом SQL для роботи з даними в базі даних. При проектуванні моделі було прийняте рішення, що даний тип даних є занадто складним для рішення поточної задачі.

Більш того, потрібні додаткові фінансові трати на хостинг серверу бази даних для забезпечення роботи з ним для усіх користувачів




| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Product Name | Sales in Each Quarter | | | | Grand Total |
| 2 | | Jan'2018 | April'2018 | July'2018 | October'2018 | |
| 3 | ABC Mutton | $ 2,667.60 | $ 4,013.10 | $ 4,836.00 | $ 6,087.90 | $ 17,604.60 |
| 4 | Crab Meat | $ 1,768.41 | $ 1,978.00 | $ 4,412.32 | $ 1,656.00 | $ 9,814.73 |
| 5 | Camembert Pierrot | $ 3,182.40 | $ 4,683.50 | $ 9,579.50 | $ 3,060.00 | $ 20,505.40 |
| 6 | Ipoh Coffee | $ 1,398.40 | $ 4,496.50 | $ 1,196.00 | $ 3,979.00 | $ 11,069.90 |
| 7 | Hot Pepper Sauce | $ 1,347.36 | $ 2,750.69 | $ 1,375.62 | $ 3,899.51 | $ 9,373.18 |
| 8 | Hot Spiced Okra | $ 1,509.60 | $ 530.40 | $ 68.00 | $ 850.00 | $ 2,958.00 |
| 9 | Mozzarella di Giovanni | $ 1,390.00 | $ 4,488.20 | $ 3,027.60 | $ 2,697.00 | $ 11,602.80 |
| 10 | Sir Rodney's Scones | $ 1,462.00 | $ 644.00 | $ 1,733.00 | $ 1,434.00 | $ 5,273.00 |
| 11 | Steeleye Stout | $ 1,310.40 | $ 1,368.00 | $ 1,323.00 | $ 1,273.50 | $ 5,274.90 |
| 12 | Veggie-spread | $ 3,202.87 | $ 263.40 | $ 842.88 | $ 2,590.10 | $ 6,899.25 |
| 13 | | | | | | |

Рисунок 2.3 – приклад excel файлу



| | update_time | qr | is_cart | is_internal | payment_terms | lead_time | incoterms |
|---|---|---|---|---|---|---|---|
| 3 | 2018-08-25 00:50:17 | | [ ] | [ ] | Net 45 | 42 Days | EXW |
| 4 | 2019-05-22 20:06:45 | 16545262a4bdee40b6a04082bae10a72c84d55fb.pn | [ ] | [v] | NET 15 | | |
| 5 | 2018-12-19 01:13:38 | | [ ] | [ ] | | | |
| 6 | 2019-06-04 20:48:39 | 35156f7099f983d1368d639a12d69b4f54f94dde.pns | [ ] | [v] | | | |
| 7 | 2019-04-07 04:47:43 | | [ ] | [ ] | Net 45 | 42 Days | EXW |
| 8 | 2019-04-03 02:46:16 | | [ ] | [ ] | | | |
| 9 | 2019-03-21 23:57:40 | | [ ] | [ ] | NET 15 | | |
| 10 | 2019-02-04 21:50:30 | | [ ] | [ ] | NET 15 | | EXW |
| 11 | 2019-01-16 01:27:37 | | [ ] | [ ] | | | |
| 12 | 2018-12-19 01:22:25 | | [ ] | [ ] | Net 30 | 6 Weeks | EXW |
| 13 | 2018-12-19 01:21:35 | | [ ] | [ ] | 45 Days | 42 Days | EXW |
| 14 | 2018-12-19 01:09:14 | | [ ] | [ ] | | | |
| 15 | 2018-12-19 01:09:00 | | [ ] | [ ] | | | |
| 16 | 2018-12-19 01:05:41 | | [ ] | [ ] | | | |
| 17 | 2019-04-08 22:32:27 | | [ ] | [ ] | NET 30 | 30 | EXW |
| 18 | 2019-05-20 17:20:47 | 42ede6bcb7ceea4a76500f9640a9432ca963a05c.pn | [ ] | [ ] | NET 30 | 30 | EXW |
| 19 | 2019-04-06 00:58:56 | | [ ] | [ ] | | | |
| 20 | 2019-04-05 22:03:14 | | [ ] | [ ] | Net 15 | 2 Weeks | EXW |
| 21 | 2019-03-26 02:05:52 | | [ ] | [ ] | NET 15 | | |
| 22 | 2019-03-25 21:18:39 | | [ ] | [ ] | Net 15 | 2 Weeks | EXW |
| 23 | 2019-03-25 19:16:54 | | [ ] | [ ] | | | |
| 24 | 2019-03-23 02:19:23 | | [ ] | [ ] | NET 30 | | |
| 25 | 2019-02-28 22:18:29 | | [ ] | [ ] | Net 15 | 2 Weeks | EXW |
| 26 | 2019-02-27 21:55:33 | | [ ] | [ ] | NET 15 | | Normal |
| 27 | 2019-02-20 20:31:19 | | [ ] | [ ] | | | |
| 28 | 2019-02-05 00:04:53 | | [ ] | [ ] | Net 15 | 2 Weeks | EXW |
| 29 | 2019-01-16 02:25:26 | | [ ] | [ ] | Net 15 | 2 Weeks | EXW |
| 30 | 2019-01-12 01:41:49 | | [ ] | [ ] | | | |
| 31 | 2021-11-18 12:40:08 | [NULL] | [ ] | [ ] | NET 15 | 7-14 Days | EXW |
| 32 | 2018-11-27 00:16:16 | | [ ] | [ ] | NET 15 | | |
| 33 | 2021-09-02 21:29:29 | [NULL] | [ ] | [ ] | NET 15 | 7-14 Days | EXW |
| 34 | 2018-11-16 02:16:02 | | [ ] | [ ] | Net 15 | 2 Weeks | EXW |
| 35 | 2018-11-14 20:49:14 | | [ ] | [ ] | NET 30 | | |
| 36 | 2018-10-24 03:55:54 | | [ ] | [ ] | | | |
| 37 | 2018-10-23 20:21:47 | | [ ] | [ ] | NET 30 | 28 | EXW |
| 38 | 2018-10-23 01:54:53 | | [ ] | [ ] | | | |
| 39 | 2018-10-23 01:38:03 | | [ ] | [ ] | | | |

Рисунок 2.4 – приклад бази даних

### 2.3 Реалізація кластеризація на Python

Використавши Python, був отриманий короткий та змістовний код алгоритму k-середніх. Ця мова була створена таким чином, щоб з нею було зручно працювати людям, що не програмували до цього. Велика купа конструкцій має близький до людської мови вигляд. Це зробило даний алгоритм простим і швидким в роботі [**Error! Reference source not found.**].

Код методу k-середніх реалізований наступними власними функціями. Лістинги цих функцій також представлені у Додатку Г:

1. Функція `define_centroids` визначає значення початкових центроїдів набору даних. Програма містить константу `CLUSTERS_COUNT=3`, що дорівнює кількості початкових кластерів, на які буде розділений набір інформації. Дана зміна є константною, тому що потрібно отримати 3 результуючі кластери (рекомендовані, нейтральні та нерекомендовані

акції). Зміна даної змінною не бажана, так як вона може призвести до неочікуваних результатів

2. Алгоритм визначає випадкові значення, що формуються в список, який далі представлений списком перших центроїдів
3. Функція `assignment` відносить дані до найближчого кластеру базуючись на квадратичній відстані. Спочатку функція знаходить відстань між кожним значенням даних та кожним центроїдом, а потім алгоритмічно визначає до якого з них це значення ближче
4. Функція `refresh_centroids` визначає нові значення центроїдів попередньо утворених кластерів. Центроїди кластерів намагаються зміститися якомога ближче до середнього значення відносно усіх значень центроїда. Після виконання цієї функції кожен центроїд опиняється майже в середині власного кластеру. Далі алгоритм буде перевіряти чи продовжують центроїди переміщуватися і, коли вони зупиняються, тоді алгоритм завершує свою роботу
5. Функція `do_clustering` оперує функціями 1-4 порядком, що передбачений методом k-середніх та описаний в розділі 1.2 даної кваліфікаційної роботи. Після виконання функції `do_clustering`, вона повертає об'єкт DataFrame, що є частиною бібліотеки Pandas, що працює з DataMining. Цей об'єкт утримує кластеризовані дані, які далі будуть конвертовані системою у результуючий excel файл

### 2.4 Хостинг системи

Яка б не була корисна та якісна система, всі зусилля не мають сенсу, якщо нею не можуть користуватися кінцеві користувачі. Доступ до системи із різних куточків світу можна забезпечити лише через Інтернет через протокол

HTTP/HTTPS. Для цього систему потрібно перемістити із свого комп'ютеру на так званий хостинг [**Error! Reference source not found.**].

Хостинг (англ. hosting) — послуга надання ресурсів для розміщення інформації на сервері (зазвичай Інтернет). Зазвичай послуга хостингу входить до пакета обслуговування сайту і передбачає, як мінімум, розміщення файлів сайту на сервері, на якому запущено ПЗ, необхідне для обробки запитів до цих файлів (веб-сервер). Як правило, обслуговування вже входить надання місця для поштової кореспонденції, баз даних, DNS, файлового сховища на спеціально виділеному файл-сервері тощо, а також підтримка функціонування відповідних сервісів. Хостинг бази даних, розміщення файлів, хостинг електронної пошти, послуги DNS можуть надаватися окремо як самостійні послуги або входити в комплексну послугу.

Існують рішення для хостингу такі як:

- Back4App
- AWS & Azure & Google Cloud
- Dokku
- Firebase
- OpenShift
- Engine Yard
- Netlify
- Heroku

Всі ці рішення мають переваги та недоліки. Після їх аналізу було прийняте рішення розмістити систему, використавши засоби Heroku, тому що цей хостинг має наступні переваги:

- Пропонує єдиний білінг для всіх проектів із розбивкою по командам Моніторинг та підвищення продуктивності завдяки розширеному моніторингу додатків
- Високопродуктивна інтеграція з Salesforce Data Mining
- Проста горизонтальна та вертикальна масштабованість
- Провідна екосистема інструментів та сервісів платформи
- Швидке управління життєвим циклом додатків та дозволами
- Пропонує потужну панель приладів та інтерфейс командного рядка.
- Інтегрується зі знайомими робочими процесами розробника
- Ряд автоматизованих функцій, включаючи масштабування, конфігурацію, налаштування та інші.
- Проста інтеграція з іншими продуктами AWS
- Гнучкість для налаштування та підтримки унікальних потреб робочого процесу DevOps

Для того, щоб змусити комп'ютерну модель працювати на Heroku хостингу, потрібно виконати наступні кроки:

1) Створити профіль на heroku.com
2) Встановити Heroku CLI на комп'ютер
3) Виконати `heroku login` та `heroku create`, щоб створити аплікацію на хостингу. Результасst команд зображений на Рис. 2.5
4) Далі зв'язуємо систему та git адресу Heroku додатку
5) Виконавши `git add * && git commit -m "First deploy" && git push heroku master` переміщуємо код системи на хостинг та запускаємо її

6) Система зараз на хостингу і до неї можна звернутися через Telegram месенджер. На Рис. 2.6. показана створена аплікація

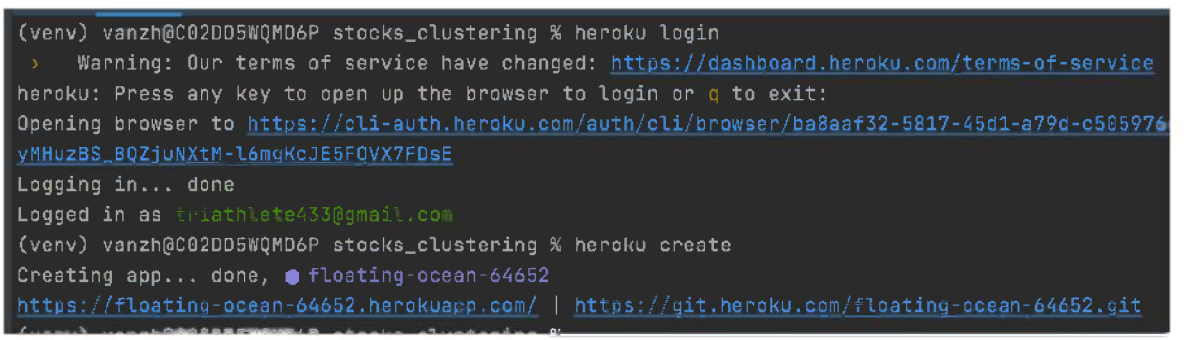



Рисунок 2.5 – початок роботи з Heroku CLI

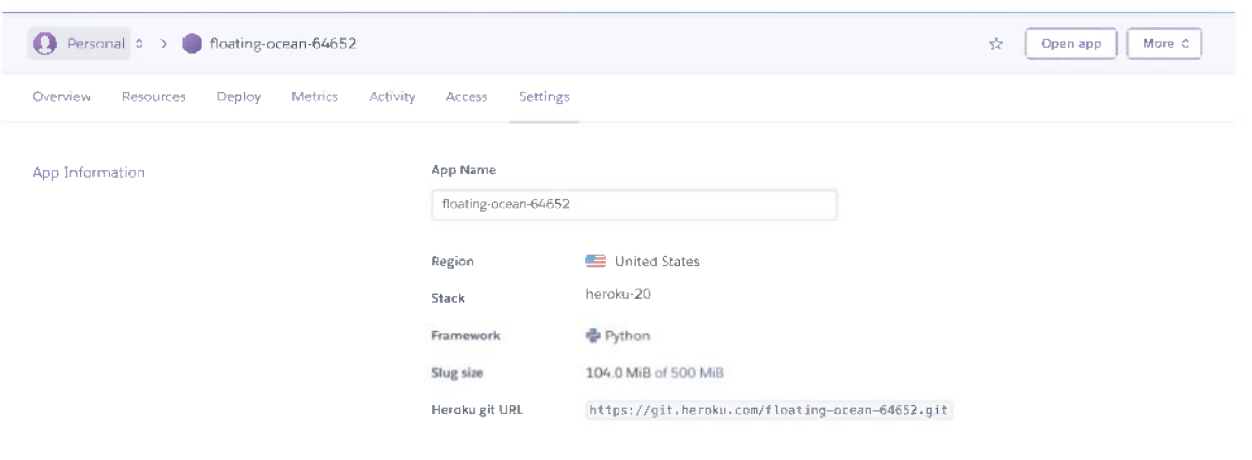



Рисунок 2.6 – аплікація на Heroku

**Висновки до розділу 2.**

В даному розділі були розглянуті та проаналізовані основні поняття та концепції створення комп'ютерної моделі аналізу індексів акцій на фондовому ринку. Огляд торкнувся методології SOLID, діаграми моделі IDEF0, а також окремо кожного функціонального модуля

Можна зробити висновок, що розроблена модель відповідає всім вимогам до комп'ютерних моделей за критеріями дизайну архітектури, розбиття моделі на функціональні модулі, проведення кінцевого тестування

# РОЗДІЛ 3.
# ІНСТРУКЦІЯ ДЛЯ КОРИСТУВАЧА

## 3.1 Мінімальні вимоги до ОС

Мінімальні вимоги до комп'ютерної моделі представлені технічними засобами системи та операційною системою, де вона працює. Python та інші засоби, використані системою, не є занадто вимогливими навіть для слабких машин. Тим паче, будь-яка ОС має вбудований функціонал для роботи з каскадними таблицями, а отже, можна не зважати уваги на розмір вихідного файлу при аналізі його користувачем. Нижче наведені вимоги, що задовольняють технічні засоби даної системи [**Error! Reference source not found.**]:

- Процесор: Intel Celeron
- Вільне місце на жорсткому диску: < 100mb
- Операційна система: Windows 7 або новіша, MacOS, Linux
- Оперативна пам'ять: 1Gb

Проаналізувавши вимоги вище, можна зазначити, що навіть застарілі комп'ютери здатні розгорнути комп'ютерну модель даної кваліфікаційної роботи [**Error! Reference source not found.**].

## 3.2 Робота системи

Основний функціонал, що представляє роботу системи, представлений у файлі app.py (Див. Додаток Г)

Шляхом HTTPS протоколу додаток отримує рік роботи фондового ринку для аналізу ринку саме в цьому році. Далі відбувається валідація року. Він повинен бути:

1. Числом

2. Значенням 1970-поточний рік. Саме в 1970 біржа почала роботу з індексом P/E

Якщо валідація не пройшла, користувач отримує відповідне пояснення і програма завершує свою роботу (Див. Рис. 3.1)


Рисунок 3.1 – помилка валідації

Якщо програма валідує дані вірними, починається збір даних фондового ринку шляхом використання відповідних API. Він розділений на 2 етапи:

1) Збір тикерів компаній (ідентифікаторів)
2) Збір P/E значень компанії

Далі починається процес кластеризації зібраних даних і формування вихідного файлу, після чого він відправляється назад користувачеві на його запит.

Дану модель можна запустити наступними шляхами:

1) Запуск на власному комп'ютері через:
    a. Командну строку. Для цього потрібно мати встановлений Python інтерпретатор на машині. Далі потрібно перейти до директорії з основним файлом програми і виконати відповідну команду запуску скрипту [**Error! Reference source not found.**]. Для забезпечення можливістю користування системи іншими користувачами, необходно передавати цілий проект на інші комп'ютери тим чи іншим чином. Сумуючи це, робимо висновок, що даний спосіб є незручним для користування
    b. Виконавчий файл. Мова Python має засоби для упаковки цілого проекту в один виконавчий файл з розширенням, залежним від ОС користувача. Даний спосіб також не є дуже зручним через те, що повинно узнавати ОС користувача і формувати відповідний файл саме для цього користувача
2) Розмістити систему на хостингу. Даний спосіб показав себе найзручнішим через наступні причини:
    a. Незалежність від типу ОС користувача

b. Незалежність від ресурсів ОС користувача, так як всі процеси працюють на потужній системі хостингу

c. Деякі хостингу не беруть плату за розміщення програми з невеликою кількістю користувачів

### 3.3 Результат роботи системи

В цьому пункті показаний приклад роботи комп'ютерної моделі та результати її роботи. Запустимо систему на дамо їй на вхід поточний рік роботи фондового ринку (Див. Рис. 3.2)

Показниками ефективності роботи моделі були вибрані час та точність даних. Час є основним фактором роботи на фінансовій біржі, але швидкість роботи не повинна компенсуватися неточними або, взагалі, втраченими даними

Так як аналогами моделі цієї кваліфікаційної роботи є Interactive Brokers та Ukraine Finance, порівняння проводяться з ними. Середній час роботи системи складає 3 хвилини, що є оптимальним результатом для складного процесу, що обробляє велику кількість інформації. Через приблизно 3 хвилини користувач отримує вихідний файл (Див. Рис. 3.3)

Вихідний файл містить 3 кластери, підкрашені відповідними кольорами. Зелений – рекомендовані компанії до інвестування, жовті – залишені на розсуд інвестора, червоні - не рекомендовані до розгляду інвестором. Більш детальний зміст вихідного файлу наведено в п. 1.6

Беручі в увагу результати використання аналогів, можна сказати, що модель виявилася більш ефективною, тому що результати аналізу співпадають на 95%, але час виконання дорівнює 3 хвилини, в той час коли результат роботи аналогів складає близько 15 хвилин

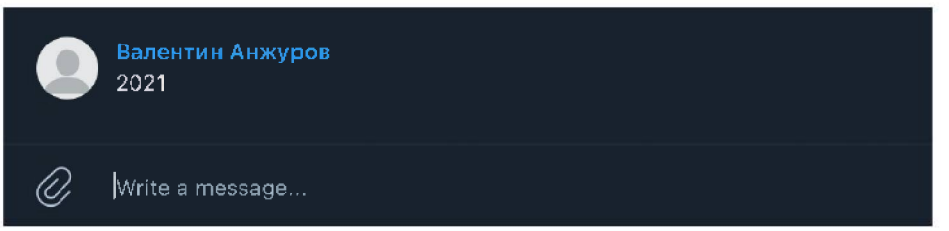



Рисунок 3.2 – передача вхідного значення

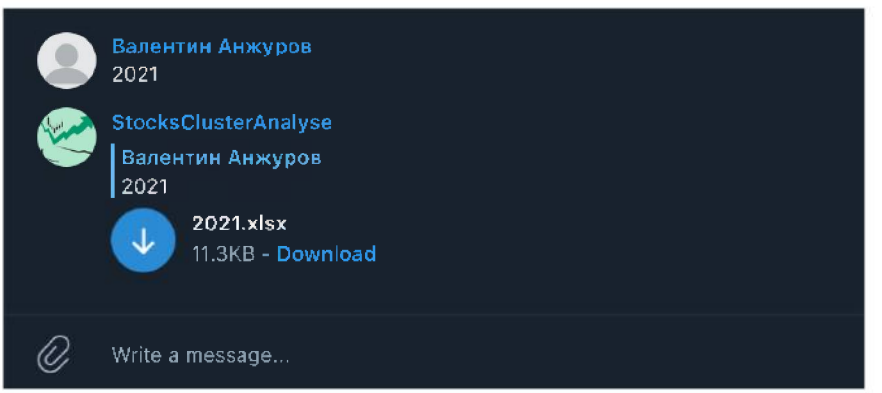



Рисунок 3.3 – отримання вихідного файлу

**Висновки до розділу 3**

В даному розділі розглянуті мінімальні системні вимогу, яким потрібна відповідати машина, на якій працюватиме комп'ютерна моделі аналізу індексів акцій на фондовому ринку.

Також, детально розписана послідовність дій, які виконує моделі для проведення.

Розписана інструкція для користувача, що матиму змогу проводити аналіз індексів

Сумуючі, можна зробити висновок, що моделі працює оптимальним шляхом та є дуже ефективною, беручі в увагу час, за який проводиться аналіз

# ВИСНОВКИ

В першому розділу були розглянуті основні поняття Data Mining, методи препроцесінгу та більше детально розглянутий метод кластериазіції – метод k-середніх. Додатково, були розглянуті використані засоби для створення комп'ютерної моделі

В другому розділу були розглянуті та проаналізовані основні моменти створення комп'ютерної моделі препроцесінгу, а саме її структуру, призначення кожної з частин моделі, структуру вхідних та вихідних даних.

В третьому розділі було описане створеної моделі та інструкція для користувача, наведені мінімальні системні вимоги моделі. Також були розглянуті проміжні та кінцеві результати роботи моделі

В ході даної роботи були розглянуті концепції препроцесінгу Data Mining, була розроблена модель препроцесінгу даних. Вивчаючи теоретичний матеріал, стало зрозуміло, що дана галузь є дуже перспективною та має право на те, щоб приділити їй увагу. Що стосується самої моделі, то вона є достатньо оптимізованою та придатною для розширення. На момент завершення роботи, модель може кластеризувати дані про студентів та виводити проміжну інформацію про стани кластерів. Серед можливостей покращення:

- Розпаралелювання алгоритму обробки даних, що дасть відчутний виграш у часі.
- Переробка моделі в універсальну. Можна зробити модель такою, що вона буде кластеризувати дані незалежно від області цих даних. Вона буде приймати на вхід таблицю з даними та назву колонки, дані з якої будуть оброблені методом кластеризації

Сумуючи, можна сказати, що мета роботи повністю досягнута. Методи були розглянуті, виявлені їх сильні та слабкі сторони.

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Что такое Data Mining: [Електронний ресурс] // Mad Data, 2019, URL: https://maddata.agency/mad-blog/chto-takoe-data-mining (Дата звернення: 13.10.2021)
2. Data Preprocessing in Data Mining: [Електронний ресурс] // Geek Forces, 2019, URL: www.geeksforgeeks.org/data-preprocessing-in-data-mining/ (Дата звернення: 10.10.2021)
3. Real-World Machine learning: [Електронний ресурс] // Manning Free Content Center, 2015, URL: https://freecontent.manning.com/real-world-machine-learning-pre-processing-data-for-modeling/ (Дата звернення: 15.10.2021)
4. Data Mining – Cluster Analysis: [Електронний ресурс] // Tutorials Point, 2019, URL: www.tutorialspoint.com/data_mining/dm_cluster_analysis.htm (Дата звернення: 21.10.2021)
5. Clustering methods Data Scientists need to know: [Електронний ресурс] // Toward Data Science, 2018, URL: https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68 (Дата звернення: 27.10.2021)
6. Pros and Cons of K-means Clustering: [Електронний ресурс] // Pros And Cons, 2020, URL:https://www.prosancons.com/education/pros-and-cons-of-k-means-clustering/ (Дата звернення: 30.10.2021)
7. K-means Clustering: [Електронний ресурс] // Toward Data Science, 2018, URL: www.towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a (Дата звернення: 07.11.2021)
8. K-means Advantages and Disadvantages: [Електронний ресурс] // Clustering in Machine Learning, 2019, URL: https://developers.google.com/machine-

learning/clustering/algorithm/advantages-disadvantages (Дата звернення: 09.11.2021)

9. Python popularity chart: [Електронний ресурс] // Google Trends, 2020, URL: https://trends.google.com/trends/explore?date=today%205-y&q=%2Fm%2F05z1_,%2Fm%2F07sbkfb,%2Fm%2F02p97,%2Fm%2F0jgqg (Дата звернення: 12.11.2021)
10. Python vs Other programming languages: [Електронний ресурс] // STX Next, 2017, URL: https://stxnext.com/python-vs-other-programming-languages/ (Дата звернення: 22.11.2021)
11. Pandas library: [Електронний ресурс] // DataFlair, 2018, URL: https://data-flair.training/blogs/advantages-of-python-pandas/ (Дата звернення: 25.11.2021)
12. Microsoft Excel для начинающих: [Електронний ресурс] // Office Guru, 2018, URL: http://www.office-guru.ru/excel/microsoft-office-excel-chto-eto-59.html (Дата звернення: 28.11.2021)
13. Мартин Р.,Мартин М. Принципы, паттерны и методики гибкой разработки на языке C#. М. : «ООО Символ Санкт-Петербург Москва», 2011. 139с.
14. Черемных С. В., Семенов И. О., Ручкин В. С. Меделирование и анализ систем. IDEF-технологии. М. : «Финансы и статистика», 2006. 6с.
15. Data Serialization: [Електронний ресурс] // Guide to Python, 2019, URL: https://docs.python-guide.org/scenarios/serialization/(Дата звернення: 30.11.2021)
16. Database server: [Електронний ресурс] // Wikipedia, 2020, URL: https://en.wikipedia.org/wiki/Database_server (Дата звернення: 30.11.2021)
17. McKinney W. Python for Data Analysis. M. : «O`REILLY», 2012. 115c.

18. Parsingin Python: [Електронний ресурс] // Software Architect, 2017, URL: https://tomassetti.me/parsing-in-python/ (Дата звернення: 30.11.2021)

19. Minimum system requirements for Python programming: [Електронний ресурс] // Quora, 2019, URL: https://www.quora.com/What-are-the-minimum-hardware-requirements-for-python-programming (Дата звернення: 20.10.2021)

20. Using Python on Windows: [Електронний ресурс] // Python docs, 2019, URL: https://docs.python.org/3.3/using/windows.html (Дата звернення: 15.10.2021)

21. Create executable from python script: [Електронний ресурс] // Data to Fish, 2020, URL: https://datatofish.com/executable-pyinstaller/ (Дата звернення: 16.10.2021)

22. S&P500 компанії: [Електронний ресурс] // Investopedia, 2019, URL: https://www.investopedia.com/terms/s/sp500.asp (Дата звернення: 17.10.2021)

23. PE index: [Електронний ресурс] // Investopedia, 2019, URL: https://www.investopedia.com/terms/p/price-earningsratio.asp (Дата звернення: 17.10.2021)

# ДОДАТКИ

**Додаток А**

## ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) Магістр
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 –Комп'ютерна інженерія

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри теоретичної та прикладної системотехніки**
**д.т.н., проф. Шматков С. І.**

«___» _____ 20___ року

### З А В Д А Н Н Я
**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

**Анжурова Валентина Євгеновича**
(прізвище, ім'я, по батькові студента)

1. Тема роботи **«Комп'ютерна модель аналізу індексів акцій на фондовому ринку»**

керівник роботи **Толстолузька Олена Геннадіївна, д.т.н., с.н.с.**,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "10" 03 2021 року № 0210-05/1804

2. Строк подання студентом роботи 19.11.2021

3. Перелік питань, які потрібно розробити)
   1. Аналіз класичних збірників літератури про фондовий ринок, його процеси та чинники впливу
   2. Дослідження існуючих індексів акцій
   3. Вивчення та практика роботи клієнта та сховища даних про поточні показники акцій ринку
   4. Побудова моделі, що працює на основі мови Python та TelegramBotAPI
   5. Тестування та налагодження роботи побудованої моделі

4. План роботи

| № з/п | Назви етапів роботи | Термін виконання етапів роботи |
|---|---|---|
| 1. | Аналіз класичних збірників літератури про фондовий ринок, його процеси та чинники впливу | 05.10.2020 – 05.03. 2021 |
| 2. | Дослідження існуючих індексів акцій | 09.03.2021– 30.04.2021 |
| 3. | Вивчення та практика роботи клієнта та сховища даних про поточні показники акцій ринку | 03.05.2021– 02.07.2021 |
| 4. | Побудова моделі аналізу індексів | 05.07.2021– 03.09.2021 |
| 5. | Тестування та налагодження роботи побудованої моделі | 06.09.2021– 08.10.2021 |
| 6. | Розробка пояснювальної записки | 11.10.2021– 26.11.2021 |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

5. Дата видачі завдання 15.02.2020

Студент Анжуров В. Є. _____
підпис

Керівник роботи Толстолузька О. Г. _____
підпис

**Додаток Б**

Затверджую

_____

«_____»_____2021 р.

**Технічне завдання**
**на розробку програмного виробу «Комп'ютерна модель аналізу індексів акцій на фондовому ринку»**

| | | |
|---|---|---|
| 1. | Введення | Назва: «Комп'ютерна модель аналізу індексів акцій на фондовому ринку»<br>Область застосування: економічні заклади аналізу фондового ринку |
| 2. | Підстава для розробки | Навчальний план ФКН за фахом 123 – Комп'ютерна інженерія.<br>1. Завдання на кваліфікаційну роботу магістра Наказ Харківського національного університету імені В. Н. Каразіна 0210-05/1804 від 10.03.2021; |
| 3. | Призначення розробки | **Мета роботи:** підвищення ефективності аналізу індексів акцій на фондовому ринку за рахунок використання комп'ютерної моделі кластеризації даних компаній на фінансовій біржі методом k-середніх<br>**Призначення розробки:** розробка комп'ютерної моделі аналізу індексів акцій на фондовому ринку.<br>1. Створення додатку на платформі Telegram месенджеру<br>2. Модель призначена для аналізу індексів акцій фондового ринку |

| | | |
|---|---|---|
| | | 3. Заходи аналізу результатів препроцесінгу даних фондового ринку комп'ютерною моделлю |
| 4. | Технічні вимоги до програмного виробу | Програма повинна:<br>1. Представляти з себе Telegram Bot у месенджері Telegram<br>2. Надавати можливість провести аналіз індексів акцій за будь-який рік функціонування ринку<br>3. Забезпечувати пошук на аналіз даних вихідного файлу<br>4. Апаратна сумісність: надати для роботи будь-який пристрій, що підтримує Telegram месенджер<br>5. Вимоги до маркування та упаковки (не висуваються)<br>6. Вимоги до транспортування і зберігання (не висуваються)<br>7. Спеціальні вимоги (не висуваються) |
| 5. | Вимоги до програмної документації | Програмною документацією щодо розроблюваного програмного продукту вважати:<br>1. Справжнє технічне завдання на розробку програми представити в Додатку Б пояснювальної записки до кваліфікаційної роботи.<br>2. Програму і методику випробувань розробленої програми представити в Додатку В пояснювальної записки до кваліфікаційної роботи.<br>3. Опис програмного виробу, представити 3-ім розділом роботи<br>4. Лістінг програми представити у Додатку Г пояснювальної записки до кваліфікаційної роботи. |
| 6. | Техніко-економічні показники | Орієнтовна оцінка ефективності кваліфікаційної роботи: не потрібна. |

| | | | |
|---|---|---|---|
| 7. | Стадії і етапи розробки | Дата | Назва етапу |
| | | 3.09. 2021 – 10.09.2021 | Написання технічного завдання. |
| | | 13.09. 2021 – 24.09.2021 | Аналіз літератури за темою роботи, щодо існуючих засобів програмних рішень |
| | | 27.09. 2021 – 1.10.2021 | Проектування архітектури моделі |
| | | 4.10. 2021 – 15.10.2021 | Розробка моделі аналізу індексів акцій на фондовому ринку |
| | | 18.10. 2021 – 29.10.2021 | Тестування та апробація комп'ютерної моделі. |
| | | 1.11. 2021 – 19.11.2021 | Розробка інструкцій користувача. |
| | | 22.11. 2021 – 26.11.2021 | Розробка пояснювальної записки до роботи. |
| 8. | Порядок контролю імоделі | 1. Перевірка ходу розробки додатку виконувати раз в 2 тижні<br>2. Випробування програмного продукту проводити відповідно до програми та методики випробувань на базі комп'ютерного класу.<br>3. Захист розробленої моделі провести на засіданні Атестаційної комісії.<br>4. Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді в 1 примірнику на CD-R компакт-диску. | |

Виконавець

студент групи КІ- 61
Анжуров В. Є.

_____

Замовник

д.т.н., с.н.с.
Толстолузька О.Г

_____

**Додаток В**

Затверджую

_____

«_____»_____ 2021р.

# ПРОГРАМА І МЕТОДИКА ВИПРОБУВАНЬ

## програмного виробу «Комп'ютерна модель аналізу індексів акцій на фондовому ринку»

### 1. Об'єкт випробувань

Найменування випробуваного програмного виробу: модель аналізу індексів акцій на фондовому ринку

Область його застосування: економічні заклади аналізу фондового ринку

### 2. Мета випробування

Перевірка працездатності виробу та правильності роботи розробленої моделі. Підтвердження характеристик моделі вимогам, які сформульовані в ТЗ (Додаток Б).

### 3. Вимоги до програми

Програма повинна:

8. Представляти з себе Telegram Bot у месенджері Telegram
9. Надавати можливість провести аналіз індексів акцій за будь-який рік функціонування ринку
10. Забезпечувати пошук на аналіз даних вихідного файлу
11. Апаратна сумісність: надати для роботи будь-який пристрій, що підтримує Telegram месенджер

12.Вимоги до маркування та упаковки (не висуваються)

13.Вимоги до транспортування і зберігання (не висуваються)

14.Спеціальні вимоги (не висуваються)

**4. Вимоги до програмної документації**

Програмною документацією щодо розроблюваного програмного продукту вважати:

1. Справжнє технічне завдання на розробку програми представити в Додатку Б пояснювальної записки до кваліфікаційної роботи.
2. Програму і методику випробувань розробленої програми представити в Додатку В пояснювальної записки до кваліфікаційної роботи.
3. Опис програмного виробу, представити 3-ім розділом роботи
4. Лістінг програми представити у Додатку Г пояснювальної записки до кваліфікаційної роботи.

**5. Засоби випробувань**

Необхідна наявність пристрою з підтримкою Telegram месенджеру

**6. Програма і методика випробувань**

Випробування проведене в два етапи:

- ознайомчий (1-й етап);

- власне випробування програмного вироби (2-й етап).

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1) Перевірка відповідності та актуальності програмної документації за критерієм наявності зазначеної в ТЗ документації.

2) Перевірка якості документації на відповідність до стандартів ЕСПД.

Перелік перевірок, що проводяться на 2 етапі випробувань, включає:

1) Перевірку відповідності технічних характеристик програми вимогам технічного завдання.
2) Перевірку ступеня виконання функціональних вимог до програми.
3) Методику проведення перевірок:

3.1) Встановити програму для роботи з Excel

Даний тест вважається пройденим, якщо серед програм є Excel (Рис. 1)

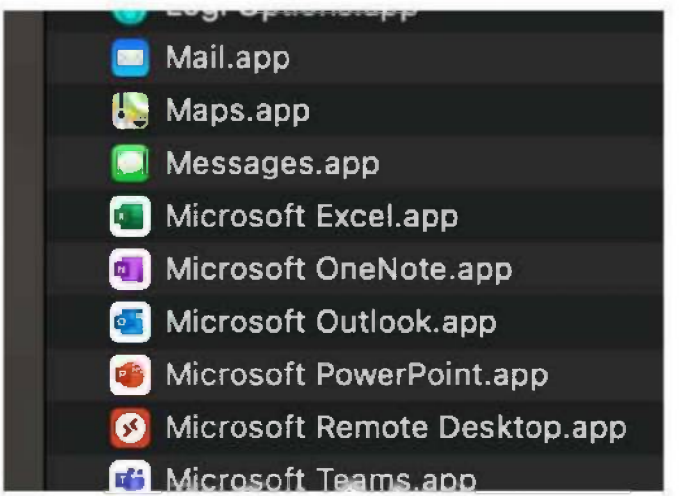



Рис. 1 – наявність Excel серед програм

3.2) Порядок проведення випробувань:

- Виконується запуск моделі. Даний тест вважати пройденим, якщо в Telegram знайдено бота (Рис. 2)

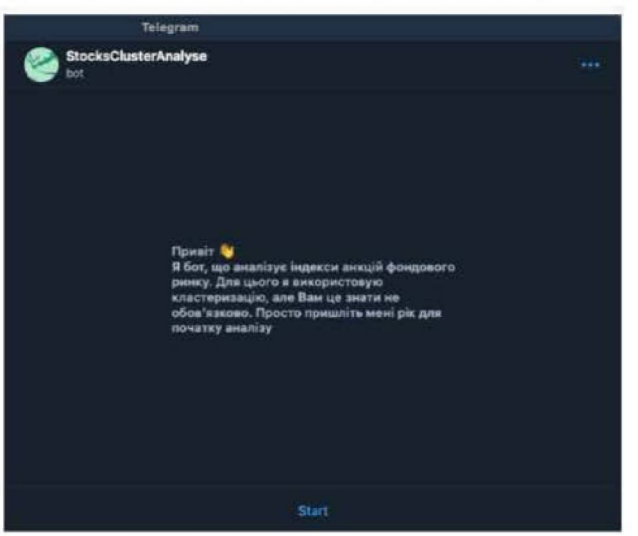



Рис. 2 – привітання бота

- На вхід подається рік роботи. Даний текст вважати пройденим, якщо не виникає помилок в консолі

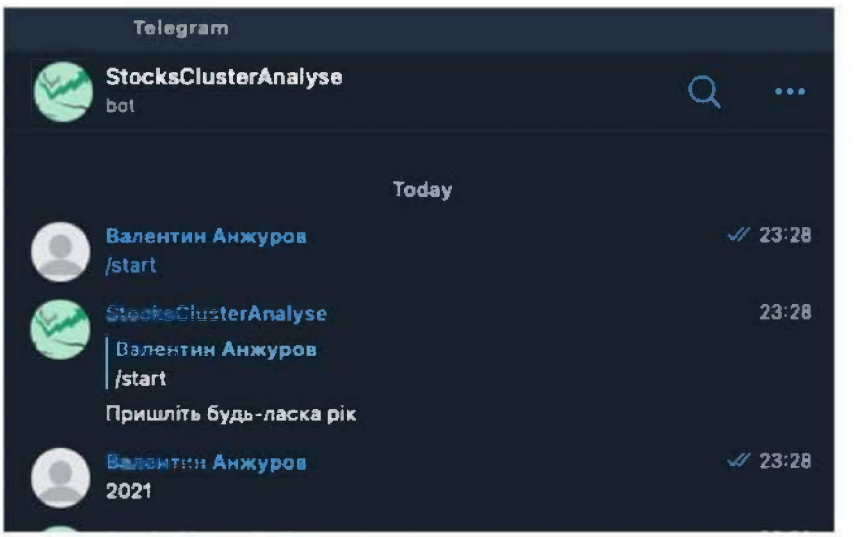



Рис.3. – успішне прийняття даних ботом

- Дані обробляються моделлю

- Створений вихідний файл повинен мати 1 сторінку з проаналізованими даними фондової біржі. Тест пройдений, якщо файл має наступний контент на Рис. 4

| | |
|---|---|
| ABC | 12,8 |
| BWA | 12,87 |
| MS | 12,95 |
| NOC | 13 |
| NWL | 13,16 |
| BK | 13,39 |
| LMT | 13,42 |
| WFC | 13,45 |
| STT | 13,48 |
| GPS | 13,79 |
| CZR | 13,99 |
| TPR | 14,16 |
| BAC | 14,21 |
| SNA | 14,54 |
| ADM | 14,64 |

Рис.4 – контент вихідного файлу

**Додаток Г**

## Лістинги коду

Лістинг Г.1 – функція, формуюча кластери

```
def cluster_data(data: list[dict], out_file_path: str) -> None:
    def _to_paint(row):
        if row.closest == 1:
            return pd.Series('background-color: green', row.index)
        if row.closest == 2:
            return pd.Series('background-color: yellow', row.index)
        else:
            return pd.Series('background-color: red', row.index)

    df = pd.DataFrame(data)
    min_pe = df["pe"].min()
    max_pe = df["pe"].max()

    centroids = _define_centroids(min_pe, max_pe)
    _assignment(centroids, "pe", df)
    while True:
        closest_centroids = df[CLOSEST_COLUMN].copy(deep=True)
        centroids = _refresh_centroids_with_column(centroids, "pe",
df)
        _assignment(centroids, "pe", df)

        if closest_centroids.equals(df[CLOSEST_COLUMN]):
            break

    sorted_pe = df.sort_values(by=["closest", "pe"])
    styled = sorted_pe.style.apply(_to_paint, axis=1)
    styled.to_excel(out_file_path,   engine='openpyxl',   index=False,
columns=["ticker", "pe"])
```

Лістинг Г.2 – функція, що визначає нові цетроїди

```
def _define_centroids(_min, _max):
    new_centroids = list(sorted(
        [random.uniform(_min, _max)
         for i in range(CLUSTERS_COUNT)]
    ))
    return dict(enumerate(new_centroids, 1))
```

Лістинг Г.3 – запуск Telegram Bot

```
server = Flask(__name__)
TOKEN = os.environ["TELEGRAM_BOT_TOKEN"]

bot = telebot.TeleBot(TOKEN, parse_mode=None)


@bot.message_handler(["start"])
def handle_text(message):
    bot.reply_to(message, "Пришліть будь-ласка рік")


@bot.message_handler(func=lambda m: True)
def handle_text(message):
```

```
    def _is_year_valid(text: str) -> bool:
        try:
            _year = int(text)
            return 1970 <= _year <= datetime.now().year
        except ValueError:
            return False

    year = message.text.strip()
    if _is_year_valid(year):
        out_file_path = run()
        with open(out_file_path, "rb") as xlsx:
            content = xlsx.read()
            bot.send_document(
                chat_id=message.chat.id,
                data=content,
                reply_to_message_id=message.id,
                visible_file_name=f"{year}.xlsx"
            )
    else:
        bot.reply_to(message, f"Можливі значення року 1970-
{datetime.now().year}")


bot.remove_webhook()
bot.infinity_polling()
```

Міністерство освіти і науки України

Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук

Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2021 р

# Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: **« Програмно-апаратна система для інтернету речей на ARM архітектурі»**

**Захищено на засіданні**
**Атестаційної комісії № 39**
протокол № __ від __.__.2021 р.
Оцінка _____ /
_____
**Голова Атестаційної комісії**
д.т.н., професор
_____**МІНУХІН**
**Сергій Володимирович**

**Виконав:**

студент 6 курсу, групи КІ– 61
Галузь знань: 12 – Інформаційні технології
за спеціальністю 123 – Комп'ютерна інженерія
**ЖИВАГА Владлен Володимирович**____

**Керівник:**

канд. тех. наук, старший викладач кафедри електроніки і управляючих систем
**МАЛАХОВА Марина Олегівна**_____

**Рецензент:**

Рецензент:
старший викладач кафедри штучного інтелекту та програмного забезпечення,
**ГУЩИН Іван Валерійович**_____

Харків – 2021

# АНОТАЦІЯ

Кваліфікаційна робота складається зі вступу, п'яти розділів, висновків, списку використаних джерел і восьми додатків. Загальний обсяг роботи становить 85 сторінок, з яких 60 сторінок основної частини з 23 рисунками та 2 таблицями, 3 сторінки списку використаних джерел із 26 найменувань, 7 додатків на 15 сторінках.

В роботі показано, що через гнучкі характеристики пристрої IoT мають вирішувати унікальні проблеми, такі як моніторинг та керування. Саме тому створення доступної та конкурентоспроможної системи дистанційного моніторингу та керування є метою цієї кваліфікаційної роботи.

У роботі був проведений аналіз. моделювання, розробка та випробування системи дистанційного керування та моніторингу. Методи, моделі та практичні рішення з впровадження апаратно-програмних засобів в системи дистанційного керування є предметом цієї кваліфікаційної роботи.

Для розробки системи були застосовані методи проектування, аналізу і моделювання систем, алгоритми збору та обробки інформації, впроваджені засоби зв'язку та системного програмування, а також адаптація ОК та МП під вимоги систем дистанційного керування.

Результатами роботи є створені цифрові інтерфейси обміну даними (I2C, SPI); система дистанційного моніторингу та керування, яка відповідає усім стандартам IoT; інтерфейс користувача у вигляді веб-додатку.

Тестування розробленої системи проводилося на промисловому приміщенні для якого важливо підтримувати постійний стан показників для зберігання. В результаті впровадження цієї системи досягнуто стабільності показників і у разі чого їх корекція автоматичним або дистанційним шляхом.

***Ключові слова:*** IOT, ДИСТАНЦІЙНА СИСТЕМА КЕРУВАННЯ, ОДНОПЛАТНИЙ КОМПЬЮТЕР, I2C, SPI

## ABSTRACT

Thesis consists of an introduction, three chapters, conclusions, list of references and eight appendices. The total amount of work is 75 pages, 50 pages of which the main part of the 18 figures and 2 tables, three page list of references with 16 titles, 7 applications at 13 pages.

The work shows that due to its flexible characteristics, IoT devices have to solve unique problems. such as monitoring and control.. The purpose of this qualification work is creating affordable, efficient, and most importantly competitive system of remote monitoring and control.

The analysis, modeling, development and testing of the remote control and monitoring system were performed. Methods, models and practical solutions for the implementation of hardware-software in remote control systems is the subject of this qualification work.

Methods of designing, analyzing and modeling systems, algorithms for collecting and processing information, implemented means of communication and system programming, as well as adaptation of SC and MP to the requirements of remote control systems were used to develop the system.

The results of the work are created digital interfaces for data exchange (I2C, SPI); remote monitoring and control system that have all IoT standards; user interface as a web application.

Testing of the developed system was carried out on an industrial premises for which it is important to maintain a constant condition of indicators for storage. As a result of implementation of this system stability of indicators is reached and in that case their correction by automatic or remote way.

***Keywords:*** IOT, REMOTE CONTROL SYSTEM, SINGLE BOARD COMPUTER, I2C, SPI

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АЦП – аналого-цифровий перетворювач.

УОМ – управляюча обчислювальна машина (пристрій).

МП – пристрій на основі мікропроцесора.

ОС – операційна система.

IoT Internet of Things Інтернет речей.

IIoT - Industrial Internet of Things – Промисловий інтернет речей.

CSS – Cascading Style Sheets - каскадні таблиці стилів.

HTML – HyperText Markup Language - мова розмітки гіпертексту.

IIoT – Industrial Internet of Things – Індустріальний Інтернет речей.

IoMT – Internet of Medical Things – Інтернет медичних речей.

SBC - ОК – одноплатний комп'ютер (Single Board Computer).

UART – Universal Asynchronous Receiver/Transmitter – універсальний асинхронний приймач/передавач.

RTOS – Real-time operating system - операційна система реального часу(ОСРЧ).

TCP/IP – Transmission Control Protocol/Internet Protocol – набір протоколів мережі Інтернет.

# ВСТУП

В наш час складно знайти людину, яка не чула про такий термін як «Інтернет речей» або IoT. Насамперед слід зазначити, що IoT не є технологією у прямому значенні цього слова. IoT – це концепція, ідея обчислювальної мережі фізичних об'єктів, що мають технології для взаємодії один з одним та/або із зовнішнім середовищем. На відміну від звичайного Інтернету, який лише спосіб об'єднання комп'ютерів, це спроба вийти за його межі. Таким чином, IoT можна назвати наступним кроком розвитку «фізичного» інтернету (маю на увазі сам Інтернет, а не Web), мета якого - об'єднати всі системи задля забезпечення додаткової вигоди для кінцевого споживача. [1]

Найпоширенішим прикладом Інтернету речей є технологія «розумний будинок». Розумними називають будинки, в яких за безпекою, комфортом та енергозбереженням стежить програмне забезпечення, що поєднує побутові прилади в єдину систему з допомогою технології передачі. [1]

Варто розуміти, що **IoT** – це системи і комплекси, що складаються з виконавчих пристроїв, розподілених та / або локальних обчислювальних ресурсів та програмних засобів, мікропроцесорів, сенсорів, програм штучного інтелекту, технологій хмарний обчислювань, передача даних між якими здійснюється за допомогою мережі Інтернет, та призначені для проведення робіт та надання послуг в інтересах користувача.[2]

За даними експертів [2] в 2025-2030 роках буде:

- 80-100 мільярдів підключень до мережі Інтернет (сьогодні – біля 16 млрд.);
- 7-19 трильйонів доларів буде складати світовий ринок IoT;
- 1 трлн. євро буде складати ринок технологій **IoT** у Європі;
- Індустрія 4.0 – дозволить отримати додатковий дохід: 30 млрд євро (Німеччина) та 110 млрд євро (Євросоюз).

Вбачається, що у 2030-2040 році технології Інтернету речей дозволять:

- 10-15% – економії бюджету на охорону здоров'я;

•10-15 років – збільшення тривалості життя;

•40-50 % – збільшення врожайності;

•15-20% збільшення пропускної здатності доріг у містах;

•85-90 % зменшення кількості автомобілів;

•10-15 разів зменшення витрат на логістику.

З врахуванням думок багатьох вчених, фахівців, експертів та на основі аналізу тенденцій розробки та впровадження технологій Інтернету речей (IP) можна стверджувати, що після 2020 року (початку офіційного запуску технології мобільної передачі даних 5G) траєкторію розвитку людства, країн, бізнесу, окремих людей може бути спрямовано у двох стратегічних напрямах: або до успіху та розквіту за умови використання технологій IoT, або до стагнації та занепаду за умови невикористання технологій IoT.

На даний час промислова галузь використовує IoT з метою підвищення своєї ефективності, покращення якості своєї продукції та / або послуг. Наприклад [3]:

- У виробництві використовують IIoT (Industrial Internet of Things) для збирання, обробки та аналізу даних з метою прогнозування, дефектування та модернізації пристроїв(обладнання), що дозволяє підприємству підвищити продуктивність роботи, знизити витрати на ремонт і оновлення обладнання, а також підвищити якість продукції.
- На державному рівні використання IoT дає можливість створення «Smart City», що передбачає регулювання трафіку, підвищення безпеки на вулицях, покращення у соціальній сфері та підвищення економічного розвитку шляхом реорганізації та економії матеріальних і природних ресурсів. Також IoT можна використовувати для підвищення якості навколишнього середовища, наприклад: збирання сміття, очищення водойм, повітря та ін.
- Технології IoT також використовуються у медицині під назвою IoMT(Internet of Medical Things). За допомогою IoMT отримуються дані для аналізу, моніторингу та прогнозуванню стану пацієнтів і людей потребуючих

догляду, що дозволяє ставити більш точні діагнози та корегувати лікування. Що призводить до скорішого одужання за менший кошт, своєчасної допомоги хворим та ін..

- Бізнес також використовує IoT з метою отримання та аналізу даних про своїх клієнтів та підприємства, що дає можливості для створення клієнтоорієнтованого продукту, підвищення ефективності на всіх етапах роботи та як наслідок зменшення витрат, що призводить до збільшення прибутків.

Головним недоліком сучасних IoT систем є необхідність величезної попередньої підготовки. Потрібно не лише навчити пристрої ідентифікувати та маркувати різні об'єкти, але й організувати спілкування пристроїв різної марки. Найчастіше гаджети сьогодні можуть взаємодіяти тільки з продукцією того ж виробника. З першого пункту випливає проблема сумісності та інтеграції даних. Сотні стандартів передачі та обробки даних обмежують можливості взаємозв'язку. Зараз все ще не встановлено, чи повинні пристрої працювати за одним стандартом чи правила треба розробляти, виходячи з конкретного постачальника інтернет-послуг.[4]

Звертаючи увагу на все сказане раніше можна зрозуміти, що введення IoT до різних сфер діяльності людини є досить *актуальною* темою. Саме тому створення доступної, ефективної, а головне конкурентоспроможної системи дистанційного моніторингу та управління є *метою цієї кваліфікаційної роботи.*

Щоб досягнути поставленої мети необхідно виконати певний перелік задач, таких як:

1. Аналіз науково-технічної літератури та інформації в Інтернеті;
2. Розробка моделі та алгоритмів окремих підсистем;
3. Аналіз апаратних компонентів та їх налаштувань до системи;
4. Розробка програмного забезпечення;
5. Проведення лабораторних випробувань системи та її модулів.

Методи, моделі та практичні рішення з впровадження апаратно-програмних засобів в системи дистанційного управління є *предметом цієї кваліфікаційної роботи*.

*Об'єктом даної кваліфікаційної роботи* є процес вирішення задач дистанційного керування шляхом розширення функціоналу одноплатного комп'ютера (ОК) та використання мікропроцесору (МП) у випадках неможливості застосування ОК. Одноплатний комп'ютер забезпечить необхідні обчислювальні можливості, збір та аналіз цифрових даних, а пристрій на базі мікропроцесору надасть можливість роботи з аналоговими сигналами. Система побудована на базі ОК та МП має широкий спектр використання та багату елементну базу.

Отже ця робота присвячена розробці системи для дистанційного управління використовуючи програмно-апаратні засоби, яка має високий обчислювальний потенціал, має широкий спектр використання та багату елементну базу.

# РОЗДІЛ 1
# АНАЛІЗ НАУКОВО-ТЕХНІЧНОЇ ЛІТЕРАТУРИ ТА ІНФОРМАЦІЇ В ІНТЕРНЕТІ

## 1.1 Ознайомлення з можливостями IoT

IoT – це концепція, яка дозволяє не тільки об'єднувати предмети матеріального світу за допомогою Інтернету для обміну інформацією між ними, але й розвивати можливості з накопичення, структурування та аналізу різної інформації про поведінку людей у міському просторі, вдома та на роботі. Виводячи на якісний інший рівень агрегацію інформації, IoT допомагає здійснити перехід від Big data до Smart data, розкриває принципово нові можливості для розвитку виробництв.[5] Але яким стандартам повинні відповідати сучасні IoT системи?

IoT системи та сучасний Інтернет мають схожу проблему – багато домашніх, корпоративних, наукових мереж об’єднаних в цілу систему, яка поєднує мережі з різними архітектурами за допомогою протоколу IP. При цьому кожний елемент має свою IP-адресу.

У IoT також одна система поєднує в собі багато не дуже гарно пов’язаних між собою елементів, кожен з яких, відповідає за окремі задачі.

Наприклад, на підприємстві створено відразу декілька мереж: системи опалювання, освітлення, кондиціювання та ін.. Різні системи (мережі) працюють за різними стандартами, тому задача їх об’єднання в одну працездатну систему досить нетривіальна. Також кожному елементу системи необхідний свій унікальний ідентифікатор. Це може бути IP-адреса, але можливості протоколу IPv4 обмежуються 4,22 мільярдами адрес, з урахуванням бурхливого зростання IoT проблема нестачі адрес може стати стримуючим фактором розвитку та поширення.

Рішенням цієї проблеми може стати використання протоколу IPv6, який дозволить кожному жителю Землі використовувати 300 млн. IP-адрес. Це дозволить майже без обмежень ідентифікувати будь-яку річ у Мережі[6].

Отже пріоритетним для систем IoT є використання технологій з підтримкою IPv6 для можливості подальшого розвитку.

## 1.2 Функціональні блоки

Система IoT складається з декількох функціональних блоків для полегшення реалізації наступних задач:

- зондування,
- ідентифікація,
- керування кінцевим пристроєм,
- зв’язок,
- управління .

На рис. 1. представлені ці функціональні блоки, як описано нижче [7]:

1. Блок «Пристрій»:

Система IoT основана на пристроях матиме можливість зондування, керування кінцевим пристроєм, контроль та моніторинг. Пристрої в такій IoT системі можуть обмінюватися даними з іншими підключеними пристроями та додатками, збирати дані з інших девайсів і обробляти їх локально, надсилати дані на централізовані сервери або хмарні додатки для обробки та безліч інших можливостей. Пристрої в IoT можуть мати декілька інтерфейсів для зв'язку з іншими пристроями, як дротовим, так і бездротовим шляхами. До них відносяться:

- інтерфейси вводу / виводу для датчиків,
- інтерфейси для підключення до Інтернету,
- інтерфейси пам'яті та зберігання,
- інтерфейси аудіо / відео.

Практично всі пристрої IoT генерують дані в тій чи іншій формі, яка при обробці системами аналітики даних генерує корисну інформацію для керування подальшими діями локально або дистанційно. Наприклад, сенсорні дані, що генеруються пристроєм контролю ґрунтової вологості в саду при обробці може допомогти у визначенні оптимальних графіків поливу.

2. Блок «зв'язок»:

Блок зв'язку здійснює зв'язок між пристроями та віддаленими серверами. Протоколи зв'язку IoT зазвичай забезпечують передачу даних на всіх рівнях комунікації між пристроями, що забезпечує цілісність та безпеку інформації.

3. Блок «служби»:

Система IoT забеспечує різні типи функцій, такі як контроль за станом пристроїв та управління ними, а також моніторинг, аналітика даних та виявлення несправних пристроїв.

4. Блок «управління»:

Блок управління забезпечує різні функції з налаштування системи IoT для пошуку оптимально режиму керування.

5. Блок «безпека»:

Блок безпеки захищає систему IoT, забезпечуючи такі функції, як аутентифікація, авторизація, конфіденційність, цілісність повідомлення, цілісність вмісту та безпека даних.

6. Блок «додаток»:

Рівень взаємодії є найважливішим для користувачів, оскільки він виступає як інтерфейс, який забезпечує необхідні модулі для управління та моніторингу різних аспектів системи IoT. Додатки дозволяють користувачам візуалізувати та аналізувати стан системи в режимі реально часу, а також прогнозуючи майбутні стани.

Отже для відповідності стандартам IoT , система повинна мати перерахований набір функціональних блоків.

## 1.3 Архітектура IoT

Розглянемо 2 типи найпопулярніших архітектур IoT:

- Тришарові (рис.1.1);
- П’ятишарові.

Тришарова архітектура відповідно має три шари:

1. Шар сприйняття (Sensor layer) – відповідальний за збір інформації та передачу інформації між об'єктами. Технологія спільної обробки інформації, технологія проміжного програмного забезпечення для збирання інформації та інші сенсорні мережі. Основна здатність цього шару реалізувати всебічне сприйняття Інтернету речей. Він є частиною Інтернету речей, включаючи ключові технології, стандартизацію та індустріалізацію, які потрібно подолати. Ключ повинен мати точніші та всебічні можливості сприйняття, а також розв’язати проблему низького енергоспоживання та невеликого розміру.[8]
2. Мережевий шар (Network layer) – відповідає за використання бездротових та дротових мереж для кодування, аутентифікації та передачі зібраних даних. Широко поширена мережа мобільного зв'язку – це інфраструктура для реалізації Інтернету речей. Це найвищий рівень стандартизації у трьох рівнях Інтернету речей. Потужна індустріалізація. Ключем до зрілої частини є оптимізація та покращення прикладних характеристик Інтернету речей та формування мережі спільного сприйняття. [8]

- Шар додатку (Application layer) – цей шар програми Інтернет речей містить бізнес користувача і функцію. Через розробку хмарних обчислень більшість систем реалізують зберігання даних та основні розрахунки у цьому шарі. [8]

Здебільшого такої архітектури замало для повноцінної роботи IoT, тому що данні системи концентруються на менших сторонах діяльності. У зв’язку з

цим набагато частіше застосовують п’ятишарову архітектуру, яка поєднує в собі 5 шарів(см. Рисунок 1) [9]:

- Сенсорний шар;
- Транспортний шар;
- Шар аналізу та обробки;
- Шар додатку;
- Бізнес-шар.

Перші два шари точно такі, як і в тришаровій архітектурі, інші шари мають наступні властивості:

- Транспортний шар (Transport layer) за допомогою таких мереж, як 3G, LAN, Bluetooth, RFID і NFC він передає данні датчика від першого шару до третього шару та навпаки.
- Шар аналізу та обробки (Processing layer) отримує дані від транспортного рівня та зберігає, аналізує і обробляє їх. Для цього він використовує такі технології, як бази даних, хмарні обчислювання та обробляючі модулі.
- Бізнес шар (Business layer) займається управлінням системою, її додатками, бізнесом та моделями прибутку. У цьому шарі здійснюється прогнозування та панування у системі.

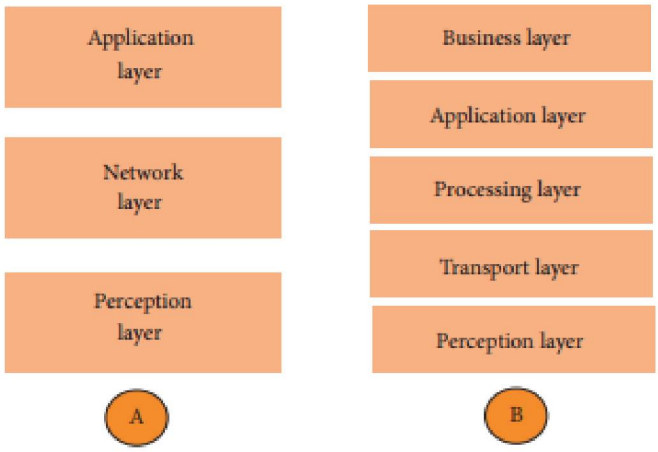



Рис.1.1. – Архітектура IoT системи: A) тришарова; B) п’ятишарова.

Отже, для ефективної подальшої роботи слід використовувати п’ятишарову архітектуру.

### 1.4 Огляд існуючих реалізацій.

Розглянемо існуючі системи дистанційного керування [10]:

1. Orvibo (рис.1.2.)

Виробник: Китай. Є англійська мова інтерфейсу. Ця система – це недорогий комплект простого в експлуатації обладнання, головне завдання якого полягає у забезпеченні безпеки будинку. І тільки в другу чергу така установка може бути базою для організації повноцінної системи "Розумний дім". Розглянемо переваги:

- простота в установці та підключенні;
- віддалений контроль через програму на смартфоні;
- автоматичне знаходження та підключення сенсорів до центрального хабу; широкий вибір пристроїв та можливість масштабування системи (близько 100 датчиків), до речі інших виробників;
- наявність власної відеокамери;
- бездротовий протокол взаємодії між контролером та датчиками (ZigBee); автономність деяких пристроїв від центрального хаба;
- вибір сценаріїв роботи з технікою будинку;
- Wi-Fi зв'язок із телефоном, обдзвін до 10 номерів;
- цілком доступна вартість (від 150 $).

До недоліків системи можна віднести:

- невелика зона дії сигналу (до 30 м);
- досить скромний набір пристроїв у базовій комплектації (тільки часткове охоплення багатокімнатної квартири чи офісу);
- відсутність резервного живлення хаба у разі відключення електроенергії;

- провідне підключення до Інтернету (для надійності роботи системи).

Можна сказати, що Orvibo Security Kit – це проміжне обладнання між класичною системою охорони приміщення та «Розумним будинком». Воно має просте та зрозуміле управління, чудові можливості для масштабування пристроями сторонніх виробників за рахунок універсального протоколу ZigBee. Однак з метою зробити цю систему доступнішою в плані покупки, тут використовуються досить примітивні датчики без захисту від злому та відключення, а камера розрахована тільки на роботу всередині приміщення.

Рис. 1.2. – Система автоматизації Orvibo

2. Xiaomi (рис.1.3.) [10]

Виробник: Китай. Є англійська мова інтерфейсу. Розумний будинок від Xiaomi відноситься до бюджетного класу обладнання, яке дозволяє зробити управління різними пристроями та побутовою технікою в будинку максимально простим та зручним. Розглянемо переваги системи:

- повна автономність пристроїв;
- можливість масштабування;
- наявність власної камери відеоспостереження;
- бездротовий протокол ZigBee;
- зручне керування за допомогою смартфона через Wi-Fi;
- наявність настроюваних сценаріїв;
- компактність та стильний дизайн; низька вартість базового комплекту (всього 90 $).

До недоліків відносяться:

- дуже невелика зона впливу сигналу (до 10 м);
- скромний набір сенсорів та виконавчих пристроїв у базовому наборі;
- на різні датчики потрібне своє становище;
- відсутність резервного харчування хаба.

Таким чином, комплект Xiaomi є чудовою стартовою платформою для підключення інших сенсорів та пристроїв, у тому числі сторонніх виробників. Її елементи працюють як самостійно, і як єдиного цілого. З їхньою допомогою можна створити досить функціональну систему контролю житлового простору, включаючи забезпечення безпеки. Дана система, враховуючи її вартість, підійде для ознайомлення із системою розумного будинку.

Рис. 1.3. – Система автоматизації Xiaomi

3. Rubetek RK-3516 (рис. 1.4.) [11]

Виробник: Росія. Є російська мова інтерфейсу . Система підтверджує, що і недорогий комплект цілком здатний вирішити проблему безпеки будь-якого житла. Інформація про кожне переміщення по території, що охороняється, миттєво передається на мобільний пристрій власника. Розглянемо переваги системи:

- сумісність із AppleHomeKit;
- голосове керування через асистент Siri;
- простота встановлення та експлуатації;
- прийнятна ціна.

Недоліком системи є те, що мобільний додаток прискорює розрядку акумулятора.


Рис. 1.4. – Система автоматизації Rubetek RK-3516

4. Vstarcam C38AR-TZ1V (рис. 1.5.) [11]

Виробник: Росія. Є російська мова інтерфейсу. Система «безпечний будинок» Vstarcam є повноцінним центром спостереження за приміщенням, здатним

забезпечити надійний захист від проникнення в будівлю сторонніх осіб. Камера, яку можна закріпити на стіні або стелі за допомогою спеціальної пластини, оснащена 1-мегапіксельною матрицею та має кут огляду 56 градусів. До переваг відносяться:

- цілодобовий online-контроль із збереженням архіву записів;
- дві антени – для Wi-Fi та сигналів датчиків;
- камера із поворотним механізмом на 360 градусів;
- датчик руху з кутом охоплення 110 градусів та робочою відстанню 8 м;
- простий та надійний дверний датчик, який спрацьовує на відкриття дверей або вікон;
- зручний пульт керування.

До недоліків можна віднести невеликий радіус дії.

Рис. 1.5. – Система автоматизації Vstarcam C38AR-TZ1V

Проаналізував існуючі рішення, зроблено висновок, що системи або дуже дорогі, або не найліпшої якості.

## 1.5 Огляд сучасних RTOS

В більшості проектів на базі IoT систем використовуються операційні системи з розподілом часу, що при збільшенні навантаження на систему або виникненні позаштатної ситуації може призвести до виходу з ладу системи, бо системи поділу часу володіють меншою пропускною спроможністю, ніж системи реального часу, так як на виконання приймається кожна запущена користувачем задача, а не та, яка "вигідна" системі, і, крім того, є накладні витрати обчислювальної потужності на більш часте переключення процесора з задачі на задачу. Критерієм ефективності систем поділу часу є не максимальна пропускна здатність, а зручність і ефективність роботи користувача. Тому використання операційної системи реального часу є більш доцільним в порівнянні з операційними системами де час розподілений.

RTOS ( Real-Time Operating System - Операційна система реального часу) - це операційна система (ОС), призначена для обслуговування додатків реального часу, які обробляють дані в міру їх надходження, як правило, без затримок у буфері. Вимоги до часу обробки (включно з будь-якою затримкою ОС) вимірюються в десяті частки секунди або більш короткі інтервали часу. Вони або управляються подіями, або розподіляються за часом.

Розглянемо головні особливості популярних операційних систем реального часу:

1.FreeRTOS

Доволі розповсюджена RTOS. Має наступні переваги:

1) Безкоштовна
2) Портована на велику кількість платформ
3) Потужний функціонал
4) Є різні бібліотеки: графіка, інтернет та інше.
5) Гарна документація.

Недоліки: доволі складна в портуванні на нові платформи

2. Keil RTX

Переваги:

1) Безкоштовна
2) Легко портується на нове залізо (у межах архітектури arm).
3) Є різні бібліотеки: графіка, інтернет та інше.

Недоліки :

1) Працювати на Keil з нею практично нереально
2) Трохи урізаний функціонал
3) Підтримується лише arm.
4) Програє багатьом ОСРВ за швидкістю.

3. QNX

Переваги:

1) Портована на велику кількість платформ
2) Легко портується на нове залізо
3) Потужний функціонал
4) Є різні бібліотеки: графіка, інтернет та інше.
5) Гарна документація.
6) Велика розробницька база.

Недоліки:

1) Комерційна, але є безкошнона ліцензія для розробки

Проаналізувавши можливості доступних RTOS, було обрану операційну систему реального часу від компанії QNX. Детальний опис системи та її переваги перед іншими системами наведено у четвертому розділі цієї кваліфікаційної роботи.

### 1.6 Постановка задачі

Після ознайомлення з науково-технічною літературою було прийнято рішення розробляти систему IoT спираючись на новітні вимоги до таких систем. Також, аби зробити систему більш доступною для користувачів були прийняті рішення щодо зменшення вартості продукту та зменшення трудовитрат на налаштування системи в місці її використання.

*Метою дослідження* є створення доступної для більшості користувачів системи, а головне ефективної та конкурентоспроможної системи IoT.

Для досягнення сформульованої мети були виділені такі *задачі дослідження:*

1. Аналіз теоретичних матеріалів щодо заявленої теми;
2. Аналіз наявних систем IoT;
3. Розробка моделі та алгоритмів окремих підсистем;
4. Аналіз апаратних компонентів та їх налаштувань до системи;
5. Розробка програмного забезпечення;
6. Проведення лабораторних випробувань системи та її модулів.

*Предметом дослідження* є практичні рішення з впровадження апаратно-програмних засобів в системи дистанційного управління.

*Об'єктом дослідження* є процес вирішення задач дистанційного керування шляхом розширення функціоналу одноплатного комп'ютера (ОК) та використання мікропроцесору (МП) у випадках неможливості застосування ОК. Одноплатний комп'ютер забезпечить необхідні обчислювальні можливості, збір та аналіз цифрових даних, а пристрій на базі мікропроцесору надасть можливість роботи з аналоговими сигналами. Система побудована на базі ОК та МП має широкий спектр використання та багату елементну базу.

# РОЗДІЛ 2. ПРОЕКТУВАННЯ МОДЕЛІ ТА АЛГОРИТМІВ ОКРЕМИХ ПІДСИСТЕМ

## 2.1 Проектування моделі.

При проектуванні моделі для IoT необхідно вирішити яка архітектура найбільше підходить для реалізації системи дистанційного керування. Ураховуючи сучасні тенденції Інтернету речей та подальший розвиток розробляємої системи, було прийнято рішення про використання п'ятишарової архітектури, яка складатиметься з шару сприйняття, танспотного шару, шару аналізу і обробки, шару додатку та бізнес шару[15]. Реалізація шарів буде виконана згідно стандартів для такого типу архітектур.

Першим архітектурним компонентом IoT є шар сприйняття. Він збирає інформацію за допомогою датчиків, які є найважливішим компонентом Інтернету речей. Існує багато типів датчиків, таких як датчик місцезнаходження (GPS), датчики руху (акселерометр, гіроскоп), камера, датчик світла, мікрофон, датчик наближення та магнітний датчик. Вони широко використовуються в різних додатках IoT. Для реалізації систем автоматичного дистанційного керування у системі повинні бути датчики для вимірювання температури, тиску, вологості, медичних показників тіла, а також ідентифікатори хімічних та біохімічних речовин. Для забезпечення функцій безпеки, система IoT повинна мати наступні елементи : ІЧ-камери, детектори руху, вимірювання відстані до об'єктів, що знаходяться поблизу, наявність диму та газів, а також датчики рівня вологи.

Отримана інформація з датчику повинна бути первинної оброблена та перетворена на дані за допомогою відповідної програми та/або алгоритму (також відомі як програми(алгоритми), що обчислюють туман). Вони переважно фільтрують та узагальнюють дані, перш ніж надсилати їх до мережі. Зазвичай такі блоки мають невелику кількість пам'яті для тимчасового зберігання, недуже потужний блок обробки та деякі функції безпеки, але

наявність таких функцій не обов'язкова. Із шаром сприйняття та первинної обробки можна на рис.2.1.

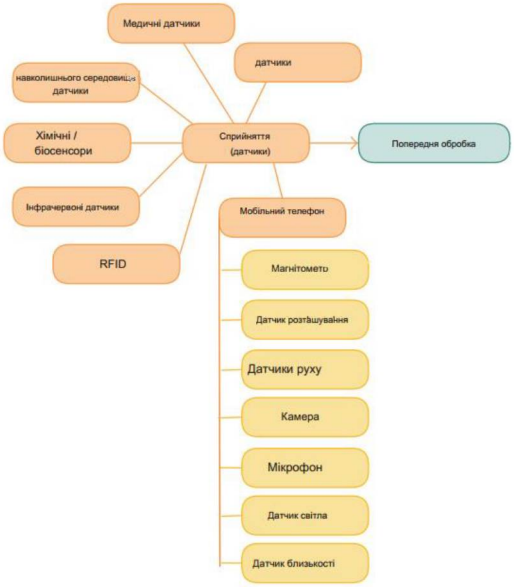



Рис. 2.1 – Структурна схема шару сприйняття та первинної обробки.

Наступна архітектурна складова – це комунікація. Різні рівні організації обмінюються даними за допомогою мережі , використовуючи різноманітні набори протоколів та стандартів. Найпоширенішими технологіями для зв'язку у мережі з низькою потужністю є RFID (радіочастотна ідентифікація) та NFC (комунікація поблизу). Для середнього діапазону це Bluetooth, Zigbee та WiFi. У разі неможливості використання бездротових мереж на допомогу приходить дротове сполучення. На рис.2.2 зображена структурна схема шару комунікації.

Комунікація у світі IoT вимагає спеціальних мережевих протоколів та механізмів, бо забезпечення якісного та доволі простого зв'язку між вузлами забезпечить стабільну роботу системи. Таким чином, були обрані та впроваджені відповідні механізми та протоколи для кожного рівня мережі згідно вимог, що висуваються для пристроїв IoT.

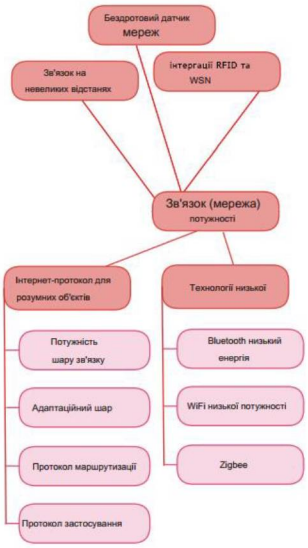



Рис. 2.2 – Структурна схема шару комунікації.

При надходженні всіх транспортованих даних до шару аналізу та обробки (рис.2.3.), вони повинні бути структуровані належним чином, перевіренні на адекватність та проаналізовані для подальших висновків, на основі яких будуть прийняті ті чи інші рішення стосовно наступних дій. Також на цьому рівні відбувається збереження даних та обробка їх для передачі до шару додатку (або програмного забезпечення).


Рис. 2.3 – Структурна схема шару аналізу та обробки.

Компоненти програмного забезпечення представлені проміжним програмним забезпеченням (далі ПЗ) та додатками. Головною метою ПЗ є створення абстракції механізму для програміста, щоб деталі апаратурної

складової відійшли на другий план та не створювали перешкод при розробці ПЗ . Такий підхід підвищить сумісність компонентів системи та спросить надання різних видів послуг. Існує багато комерційних та відкритих засобів для надання проміжних програм на пристрої IoT, наприклад: OpenIoT, MiddleWhere, Hydra, FiWare та Oracle FusionMiddleware.

Шар бізнесу відповідальний за управління всією системою IoT, включаючи додатки, бізнес і моделі прибутку, а також конфіденційність користувачів. За допомогою цього шару здійснюється прогнозування та панування у системі. Також на цьому рівні беруться дані для відповідних спеціалістів з митою розвитку системи. Цей шар також може бути об'єднаний з шаром ПЗ для простоти реалізації системи та зменшення навантаження на неї. Структурна схема шару бізнесу та ПЗ зображено на рис.2.4.


Рис. 2.4 – Структурна схема шару бізнесу та ПЗ

Для повного розуміння розробленої моделі пропонується розглянути принципову схему (рис.2. 5). На схемі зображені всі розглянуті елементи та послідовність роботи системи.

Також на цьому етапі необхідно розробити алгоритми окремих підсистем. Для реалізації системи необхідно розробити наступні алгоритми: алгоритм отримання інформації, алгоритм опитування датчиків та алгоритм перетворення аналогової інформації на цифрову.

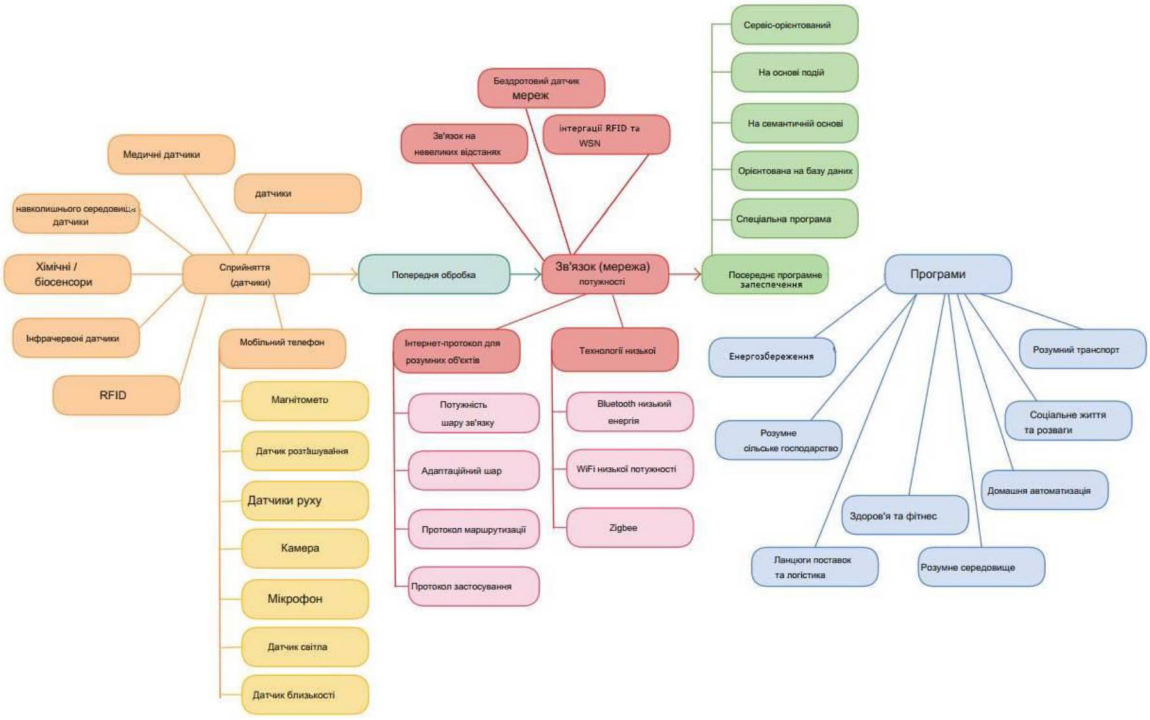



Рис.2.5 – Принципова схема моделі.

Отже розроблена модель забезпечує можливість реалізації усіх функціональних блоків та відповідає п’ятишаровій архітектурі.

## 2.2 Особливості цифрових систем збору та первинної обробки отриманої інформації.

Система збору та попередньої обробки інформації є неодмінною частиною будь-якої автоматизованої системи управління. Призначення цієї системи - отримання необхідних даних, контроль відповідності параметрів допустимим значенням, обчислення необхідних техніко-економічних показників і періодична реєстрація результатів.

На рис. 2.6. представлений приклад структурної схеми цифрової системи контролю і управління технологічним процесом[16].

Фізичні величини - температура, вологість, тиск та ін., що є об'єктом контролю в даній системі, - перетворюються за допомогою датчиків в електричні сигнали.

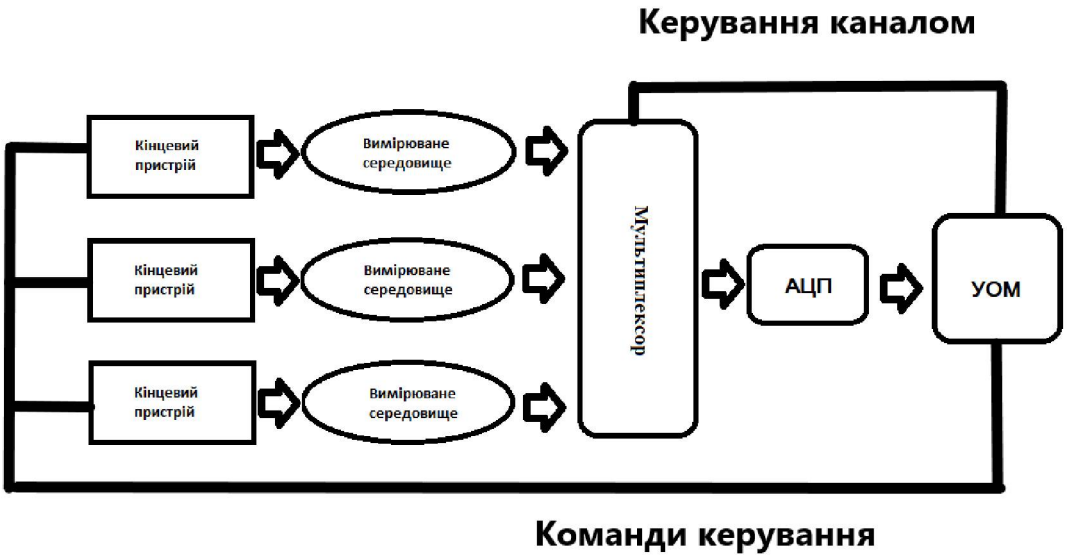



Рис. 2.6 – Структурна схема цифрової системи контролю: АЦП - аналого-цифровий перетворювач; УОМ – управляюча обчислювальна машина

Сигнали в системі використовуються двох типів: цифрові та аналогові. Відповідно до цього використовуються два типи датчиків: цифрові та аналогові. Цифрові датчики видають цифрову інформацію придатну для обробки на УОМ. Аналогові датчики видають аналогову інформацію, і тому їх можна підключати до УОМ тільки через аналого-цифровий перетворювач (АЦП). Якщо в систему контролю входить велика кількість датчиків, то доцільно використовувати мультиплексор - швидкодіючий перемикач вузлів вимірювання для обслуговування одним АЦП цілої групи датчиків. АЦП підключається безпосередньо до внутрішньої шини УОМ і стає її структурним доповненням. Виміряні величини, перетворені в цифрову форму, надходять в УОМ і обробляються відповідно до закладених в пам'ять машини програмами типових алгоритмів збору та попередньої обробки інформації. Опитування (зчитування показників) датчиків може відбуватися циклічно і ациклічні. При циклічному опитуванні порядок підключення датчиків до КОМ зберігається

постійним у часі. Ациклічне опитування вузлів вимірювання проводиться, як правило, у випадках відхилення показників від заданої норми.

Отже для підвищення швидкодії системи треба за можливості використовувати цифрові пристрої ( датчики, мікроконтролери та ін. ). А також при стандартному режимі роботи використовувати циклічний алгоритм опитування, а при появі відхилені від норми – ациклічний.

### 2.3 Процеси перетворення інформації в цифрових системах.

В процесі перетворення безперервного сигналу в послідовність цифрових даних відбувається подвійне квантування інформації: квантування за часом і квантування за рівнем (рис. 2.7. а). квантування за часом відбувається внаслідок того, що вимір відбувається не безперервно, а тільки в дискретні моменти часу. Фізично дискретізатор, або квантувач за часом, представляє собою швидкодіючий тригерний (ключовий) елемент (рис. 2.7. а). У цифрових системах інтервал часу $t_0$, через який відбувається спрацьовування тригеру, - величина постійна, яку називають по-різному: циклом опитування, тактом квантування або періодом дискретизації. На виході ключового елемента з безперервного сигналу $x(t)$ виходить дискретний сигнал $x_д(t)$, що складається з послідовності імпульсів, висота кожного з яких дорівнює величині безперервного сигналу в дискретні моменти часу (рис. 2.7. б). математично дискретна функція $x_д(t)$, що отримується шляхом квантування по часу безперервного сигналу $x_д(t)$ з постійним тактом $t_0$, описується виразами:

$$x_д(t) = x(kT0) \quad \text{при} \quad t = kT0,\ k = 0, 1, 2, \ldots;$$
$$x_д(t) = 0 \quad \text{при} \quad kT0 < t < (k + 1)T0. \qquad (2.1)$$

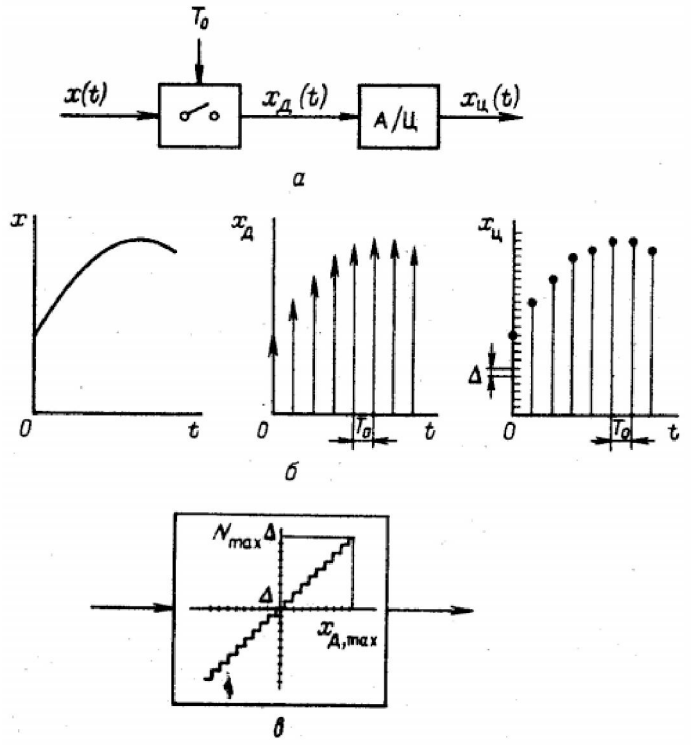



Рис. 2.7 – Формування цифрових сигналів з безперервних: а - блок-схема аналого-цифрового перетворювача; б - процеси квантування за часом і за рівнем; в - статична характеристика АЦП

У перетворювачі «аналог-цифра» значення амплітуди кожного імпульсу *хд* піддається округленню, кодується в цифровий сигнал *хц*, після чого надходить в центральний процесор УОМ. Квантування за рівнем пояснюється тим, що цифрове (чисельне) значення вимірюваної величини визначається при її зіставленні з деякою градуйованою шкалою, що складається з цілого числа дискретних них ступенів (рис. 2.7. г). Ціна поділки, або крок квантування за рівнем Δ, визначається розрядністю перетворювача «аналог-цифра», тобто числом двійкових розрядів для запису чисел. Оскільки максимальне число різних чисел, які можна представити за допомогою двійкового коду розрядністю r, дорівнює:

$$N_{max} = 2_r - 1, \qquad (2.2)$$

звідси знаходимо крок квантування за рівнем:

$$\Delta = \frac{1}{N_{max}} = \frac{1}{2^r - 1} \approx \frac{1}{2^r}$$

(2.3)

Цифровий сигнал $x_ц$ на виході АЦП формується як ціле число L кроків квантування за рівнем Δ, що містяться в кожному імпульсі $x_д$, поступає на його вхід в моменти часу t = 0, T0, 2T0, ..., тобто

$$x_ц = L\Delta,\ L = 0, 1, 2, \ldots, N_{max}. \quad (2.4)$$

Залишок $\delta_x < \Delta$ округляється, або буде скорочуватися. В обох випадках справедливе співвідношення:

$$x_д = x_ц + \delta_x, \quad (2.5)$$

де похибка квантування укладена в наступних межах:

$$-0{,}5 \leq \frac{\delta_x}{\Delta} < 0{,}5 \text{ при округлені} \quad (2.6)$$

та

$$0 \leq \frac{\delta_x}{\Delta} < 1 \text{ при скорочені} \quad (2.7)$$

Квантування за рівнем можна уявити як процес проходження сигналу $x_Д$ через елемент з багатоступеневою характеристикою, зображеною на рис. 7 *в*. Таким чином, квантування за рівнем принципово є джерелом нелінійності. Однак у АЦП, що мають не менше 10 двійкових розрядів, ефекти квантування за рівнем практично непомітні, оскільки похибка квантування за рівнем становить менше статичних і динамічних помилок датчиків. У цьому випадку вихідний сигнал АЦП $x_ц(t)$, що надходить в УОМ, можна з достатньою точністю розглядати як дискретний тільки по часу сигнал:

$x_ц(t) = x(kT_0)$ при $t = kT_0$, $k = 0, 1, 2, \ldots$;

$$x_ц(t) = 0 \qquad \text{при } kT_0 < t < (k+1)T_0. \qquad (2.8)$$

Оскільки УОМ отримує інформацію про величини, які безперервно змінюються дискретно у часі, виникає задача відновлення значень вимірюваних величин в проміжках між моментами квантування. Таке відновлення проводять різними методами екстраполяції та інтерполяції. Найпростіший метод екстраполяції - це ступінчаста екстраполяція, при якій миттєве значення дискретного сигналу $x(kT0)$ фіксується на період, рівний такту квантування (рис. 2.8. а). Така екстраполяція не вимагає ніяких обчислювань, що є її суттєвою перевагою.

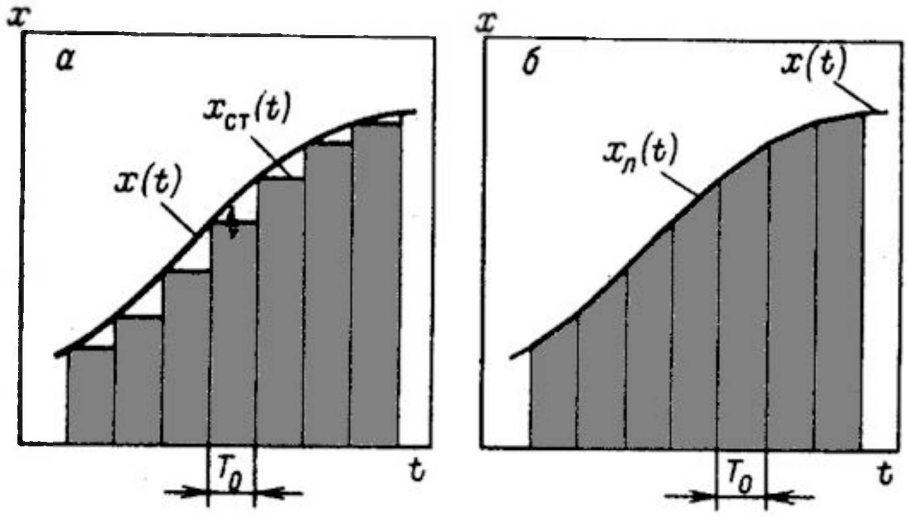



Рис. 2.8 – Способи екстраполяції дискретно вимірюваних величин: а - ступінчаста; б - лінійна

У той же час при одному і тому ж періоді квантування ступінчаста екстраполяція дає більшу похибку, ніж інші методи екстраполяції (лінійна, параболічна і ін.), Для реалізації яких, однак, потрібно виконувати певні обчислювальні операції (рис. 2.8.б). Для зниження навантаження на УОМ в системах контролю з багатьма датчиками задану точність визначення контрольованих величин зазвичай вимагається забезпечити без застосування складних алгоритмів екстраполяції. Точність вимірювання величини оцінюють середньої квадратичної похибки σ. При цьому допускається, що вимірювані величини $x(t)$ - це випадкові стаціонарні ергодичні процеси, а

похибка датчика $\Delta x_u(t)$ не залежить від вимірюваних величин. З урахуванням похибки датчика ступінчаста екстраполяція вимірюваної величини може бути записана в наступному вигляді:

$$x_{cm}(t) = x(kT_0) + \Delta x_u(kT_0) \quad \text{при} \quad kT_0 \le t < (k+1)T_0,\ k = 0, 1, 2, \ldots \qquad (2.9)$$

На інтервалі між сусідніми вимірами середня квадратична похибка ступінчастої екстраполяції є функцією часу. У моменти часу $t = kT_0$, $k = 0, 1, 2, \ldots$ ця похибка буде мінімальною, яка дорівнює середньому значенню квадратичної похибки датчика. Середня квадратична похибка зазвичай досягає максимуму в кінці періоду дискретизації. При ступінчастій екстраполяції максимальну середню квадратичну похибку знаходження величини можна розрахувати следую- чином:

$$\sigma_{\max} = 2\ K_x(0) - K_x(T_0) + \sigma_u\,, \qquad (2.10)$$

де $K_x$ - кореляційна функція вимірюваної величини; $\sigma_u$ - середня квадратична похибка датчика.

З виразу (10) випливає, що при відомих статистичних характеристиках вимірюваної величини і похибки датчика задану точність визначення можна забезпечити відповідним вибором періоду дискретизації $T_0$.

## 2.4 Алгоритм збору та попередньої обробки інформації

*Алгоритм циклічного опитування датчиків.* Як орієнтир при оцінці стану системи в першу чергу цікавиться відхиленням контрольованих параметрів від заданих значень або норм. На рис. 2.9 графічно представлений алгоритм циклічного опитування датчиків і порівняння їх показань з нормою. Передбачається, що датчики пронумеровані послідовно, починаючи з першого і і закінчуючи n-m. Вихідними даними для алгоритму слугує кількість датчиків n, масиви гранично допустимих значень датчиків і адресний масив A, B, C, D для збору даних про вихід за межі норми показників датчиків. В осередках пам'яті A, B, C, D відповідно зберігаються показання і-го датчика $x_i$,,

відхилення його значення від норми $\Delta x_i$, порядковий номер датчика $i$ та час $t_i$, в яке відбулося відхилення від норми.

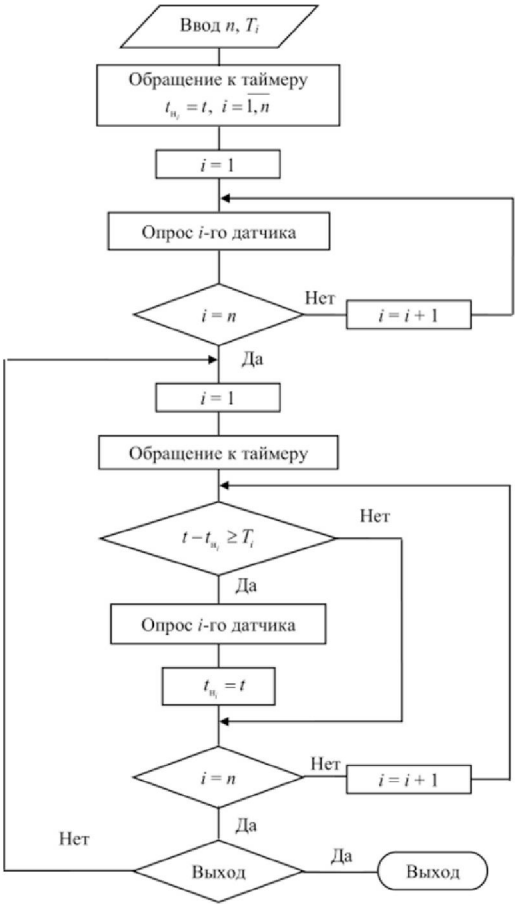



Рис. 2.9 – Блок-схема алгоритму циклічного опитування і контролю їх показань

Опитування починається з першого датчика. Спочатку слід порівняти показання датчика з верхньої нормою. Якщо це буде свідчення не виходить за верхню межу значення, переходять до порівняння з нижньої нормою. Якщо показання першого датчика не виходить за межі верхньої і нижньої норми, переходять до опитування другого датчика. Аналогічно обробляється інформація, прочитується з наступних датчиків. Процес, параметри якого не виходять за задані межі, не потребує в управлінні, тому, якщо при опитуванні всіх датчиків системи їх показання залишаються в межах норми, машина припиняє роботу з даного алгоритму, корекція управління непотрібна. Для наступного циклу опитування датчиків необхідно повторити запуск програми.

Якщо показання і-го датчика вийшли за межі норми, то відповідно до алгоритму, представленого на рис. 4 виконуються наступні операції:

- звертаються до таймера;
- в осередку пам'яті A, B, C, D записують відповідно величини $x_i$, $\Delta x_i$, $i$, $t_i$;
- за допомогою змінної j відзначають порядковий номер події виходу контрольованого параметра за межі норми;
- з ячійок A, B, C, D величини $x_i$, $\Delta x_i$, $i$, $t_i$ переписують в нові ячійки пам'яті A + 1; B + 1; З + 1; D + 1;
- ячійки A, B, C, D очищують та готують для прийому даних на випадок, якщо показники наступного датчика вийдуть за межі норми;
- проводять опитування (і + 1) -го датчика.

Після завершення циклу опитування всіх датчиків відбувається аналіз отриманих даних. Дані записуються до масиву (таблиці), яка містить j рядків і чотири стовпці з величинами $x_i$, $\Delta x_i$, $i$ і $t_i$ відповідно. На основі сформованих даних проводиться корегування стану системи.

На цьому виконання заданої програми припиняється. Через проміжок часу, що дорівнює періоду опитування, цю програму знову запускають для обробки нових даних, що надходять від вимірювальних датчиків.

Алгоритм циклічного опитування датчиків і порівняння їх показань з нормою іноді модифікують і контролюють граничні значення до декількох ділянок, відповідних, наприклад, аварійні ситуацій. У цьому випадку окрім аналізу даних одночасно користувачеві подається повідомлення про наявність проблем, наприклад, перенапруга або відчинене вікно.

Крім необхідності виявлення факту виходу контрольованої величини за межі допуску, часто виникає потреба у визначенні причини і місця несправності, що викликала порушення нормального режиму роботи. Якщо відома причина порушення, несправність можна швидко усунути. У

загальному вигляді виявлення несправності та причин порушення є завданням технічної діагностики. У простих системах, до яких відноситься ця система дистанційного керування, проблема виявлення місця і причини несправності може бути вирішена методами логічних міркувань на основі очевидних властивостей причинно-наслідкових зв'язків.

*Алгоритми визначення дійсних значень вимірюваних величин за показаннями датчиків.* Сигнал в цифровому коді, який надходить в УОМ з датчика, визначає не саму вимірювану величину, а значення вихідного сигналу датчика. У загальному випадку вихідний сигнал датчика Y пов'язаний з вимірюваною величиною $x$ нелінійної залежністю

$$Y = \varphi(x), \tag{2.11}$$

де $\varphi(x)$ - монотонна функція. Звідси істинне значення вимірюваної змінної можна отримати, вирішивши рівняння

$$x = \varphi^{-1}(y), \tag{2.12}$$

де індекс -1 означає зворотну функціональну залежність.

Залежно від виду функції $\varphi(x)$ існують різні способи вирішення цього рівняння. Найбільш простий спосіб виходить тоді, коли функція $\varphi(x)$ допускає лінійну апроксимацію виду

$$\varphi(x) = a + bx. \tag{2.13}$$

Більшість датчиків у системі мають лінійну характеристику показників. Тому с виразів (11)-(13) отримуємо:

$$x = (y - a)/b. \tag{2.14}$$

Для визначення справжнього значення вимірюваної змінної $x$ по вихідному сигналу датчика $y$ в пам'ять машини вводять постійні параметри $a$ і $1/b$ та проводять обчислення згідно з алгоритмом (14).

Якщо функція $\varphi(x)$ - нелінійна, але аналітична, справжнє значення вимірюваної змінної визначають відомими рекурентними методами. Так,

наприклад, якщо $y = x^2$, то для обчислення $x = \sqrt{y}$ використовувати рекурентну формулу:

$$x = \frac{1}{2\left[x_{n-1}+\left(\frac{y}{x_{n-1}}\right)\right]}, \quad n = 1,2,3, \ldots. \tag{2.15}$$

Послідовність $x_n$ монотонно сходиться до значення $\sqrt{y}$, причому число ітерацій тим менше, чим ближче нульове наближення $x_0$ до шуканого результату. Зазвичай за нульове наближення $x_0$ приймають грубу оцінку $\sqrt{y}$ по нестачі.

Для ряду датчиків функція $\varphi(x)$ буває задана в табличному вигляді. Інші датчики градуируют експериментально по точкам. У цих випадках для визначення істинного значення вимірюваної величини користуються лінійною інтерполяцією таблиці значень: $x1$, $y1$; $x2$, $y2$; … $x_n$, $y_n$ з заданим кроком $\Delta x_i = x_{i+1} - x_i$ по змінній $x$:

$$x = x_i + \Delta x_i^{i+1} \frac{y - y_i}{y_{i+1} - y_i}. \tag{2.16}$$

Алгоритм визначення істинного значення x містить наступні операції :

- упорядкований перебір значень $y_i$ з метою знаходження найближчої до $y$ пари значень $y_i + 1, y_i$, які відповідають умові $y_i \le y \le y_i + 1$;
- визначення $x_i$, відповідного $y_i$;
- обчислення x відповідно до вираження (2.18).

Для реалізації алгоритму в пам'ять машини необхідно записати всю таблицю значень $x1, y1; x2, y2; \ldots x_n, y_n$, а також розмір кроку $\Delta x^{i+1}$.

Недолік табличного способу - велика витрата пам'яті машини. Більш раціональний з точки зору використання пам'яті машини - спосіб визначення істинного значення $x$ шляхом апроксимації заданої залежності $x = \varphi^{-1}(y)$ за допомогою полінома ступеня $m$:

$$x = P_m(y) = a_m y^m + a_{m-1} y^{m-1} + \ldots + a_1 y + a_0 , \qquad (2.17)$$

де коефіцієнти полінома $a_j$ ($j$ =0, $m$ ) підбирають так, щоб помилка апроксимації не перевищувала допустимого значення $\delta_{доп}$.

$$\delta_i = P_m(y_i) - x_i , \; i = 1, n \qquad (2.18)$$

В цьому випадку в пам'ять машини записують тільки коефіцієнти полінома $a_j$ ($j$ =0,$m$ ).. Поліном для будь-якого значення у обчислюють за так званою схемою Горнера. Для цього поліном (17) перетворять до наступного виду:

$$P_m(y) = ((\ldots(a_m y + a_{m-1}) y + a_{m-2}) y + \ldots + a_1) y + a_0. \qquad (2.19)$$

Коефіцієнти полінома $a_j$ ($j$ =0,$m$ ) записують в пам'ять машини поспіль в порядку убування номерів їх індексів. Значення $P_m(y)$ при зазначеному порядку розташування коефіцієнтів $a_j$ обчислюють $m$ разів за рекурентною формулою

$$R_{m-j+1}(y) = R_{m-j} y + a_{j-1} \qquad (2.20)$$

до отримання

$$R_m(y) = P_m(y). \qquad (2.21)$$

*Алгоритми визначення сумарних і середніх значень вимірюваних величин.* У системах контролю та управління до режимних показників системи відносяться сумарні та середні витрати ресурсів. Для визначення сумарних показників за допомогою УОМ використовують різні методи дискретного інтегрування.

Найпростіший метод дискретного інтегрування - метод Ейлера, при якому використовується ступінчаста екстраполяція безперервної кривої підінтегральної функції. Цей метод ще називають методом прямокутників, оскільки при ньому інтеграл апроксимується сумою площ прямокутників (рис. 2.10. а). Кожен прямокутник має підставу, що дорівнює періоду дискретизації $T0$, і висоту, дорівнюючу попередньому значенню інтегрованої

функції $x[(k-1)T_0]$. Оскільки площа кожного прямокутника дорівнює $T_0x[(k-1)T_0]$ , алгоритм дискретного інтегрування можна записати у вигляді:

$$Sx(kT_0) = Sx[(k-1)T_0] + T_0\,x[(k-1)T_0], \qquad (2.22)$$

де $Sx\,[(k-1)\,T_0]$ - сума всіх площин прямокутників, виключаючи площу самого останнього прямокутника.

Перевагою алгоритму (24) є простота його реалізації, а недоліком - те, що методична похибка цього методу значною мірою залежить від виду інтегрованої функції і обраного періоду дискретизації.

Для задач, в яких потрібна вища точність інтегрування, використовують інші алгоритми дискретного інтегрування.

*Метод трапецій* (рис. 2.10. б) забезпечує більшу точність інтегрування в порівнянні з методом Ейлера, оскільки до апроксимації прямокутниками додаються трикутні елементи. Алгоритм дискретного інтегрування в цьому випадку має вигляд:

$$S_x(kT_0) = S_x[(k-1)T_0] + T_0x[(k-1)T_0] + \left(\frac{T_0}{2}\right)\{x(kT_0) - x[(k-1)T_0]\} \qquad (2.23)$$

Для оцінки точності різних методів дискретного інтегрування необхідно зіставити отримані результати з деяким еталонним обчислювальним процесом.

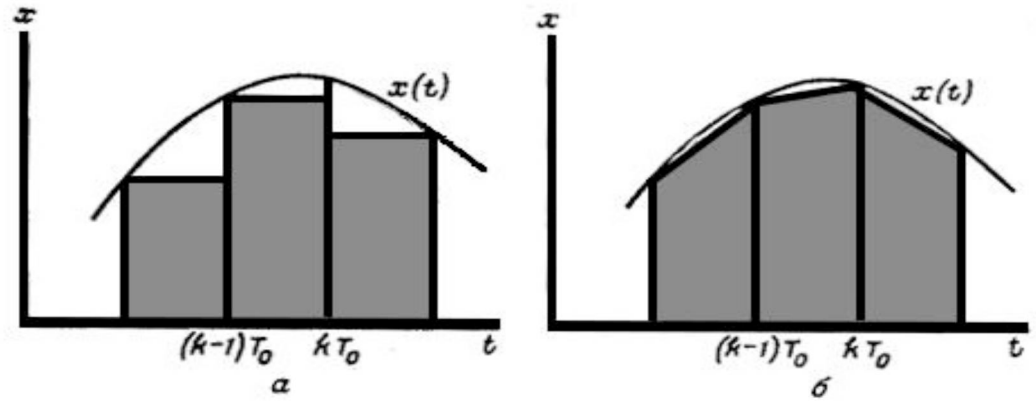



Рис. 2.10 – Області підсумовування при дискретно інтегруванні: а - метод прямокутників; б - метод трапецій

Для визначення деяких техніко-економічних показників необхідно обчислити середні значення вимірюваної величини за заданий проміжок часу $t_0$. Формування поточних середніх значень використовується також в задачах контролю та керування для згладжування вимірюваних величин. Якщо тим чи іншим методом дискретного інтегрування в системі обчислюється сумарний показник $S(t_0)$ за час $t_0 = kT_0$, то середнє значення можна обчислити за формулою:

$$x_{cp} = S(t_0)/t_0. \quad (2.24)$$

В іншому випадку для визначення середнього значення використовують алгоритм покрокового сумування усереднюються величини протягом усього інтервалу часу усереднення:

$$x_{\text{ср}} = \frac{1}{n}\sum_{i=1}^{n} x_i(t_i), \overline{1,n} \quad (2.25)$$

де n - число кроків дискретизації за часом $t_0$: $n = \frac{t_0}{T_0}$

Даний набір алгоритмів дозволить отримати якісні дані про стан навколишнього середовища, що дасть можливість більш якісно здійснювати контроль за системою та керування нею.

## 2.5 Аналого-цифровий перетворювач.

Аналого-цифрові перетворювачі (АЦП) - це вимірювальні перетворювачі, призначення яких полягає в автоматичному перетворюванні вимірюваної аналогової величини в дискретну, представлену у вигляді цифрового коду. Відповідно до методу побудови усі АЦП можна розділити на три групи: з времяімпульсним перетворенням, частотно-імпульсним перетворенням і порозрядного врівноваження. Для розробляємої системи ми використовуємо АЦП порозрядного врівноваження.

АЦП порозрядного врівноваження. Розглянемо роботу цього АЦП на прикладі перетворювача напруга-цифровий код. Структурна схема АЦП

порозрядного перетворення представлена на рис. 2.11. Вимірювана напруга $u_X$ порівнюється з набором зразкових напруг $u_{01} > u_{02} > ... > u_{0n}$ складеним за певним законом, наприклад, відповідно до розрядів двійкової системи числення. Ці напруги надходять на пристрій порівняння ПС від перетворювача код-зразкова напруга відповідно до команд пристрою керування. Перетворювачем «код-зразкова напруга» є цифро-аналоговий перетворювач (ЦАП), завданням якого є вироблення аналогової напруги відповідно до надходячого на його вхід числового коду.

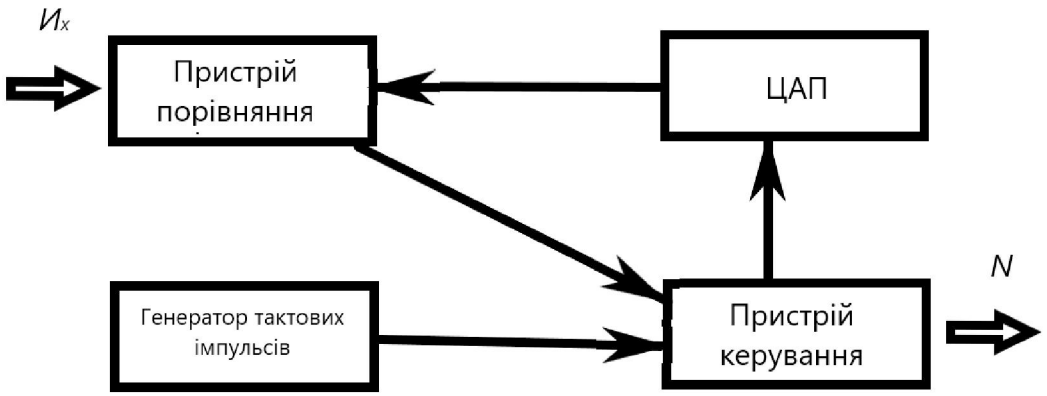



Рис. 2.11 – Структурна схема АЦП з порозрядним врівноваженням

Послідовність роботи АЦП порозрядного врівноваження задається генератором тактових імпульсів (ГТІ). У першому такті відбувається порівняння вхідної напруги $u_x$ з найбільшою зразковою напругою $u_{01}$. Якщо їх $u_x < u_{01}$, т. е. $u_x - u_{01} < 0$, то пристрій управління подає на вихід код 0. Таким чином, вищий розряд вихідного двійкового коду буде нульовим. Після цього напруга $u_{01}$ відключається від пристрою порівняння та подається напруга $u_{02} = u_{01} / 2$. Після цього знову відбувається порівняння, на цей раз $u_x$ та $u_{02}$. Якщо знову $u_x - u_{02} < 0$, то знову від пристрою порівняння відключається $u_{02}$ і надсилається 0 в наступний розряд двійкового коду. Це відбувається до тих пір, поки не буде $u_x - u_{0i} > 0$. Тоді $i$-му розряду буде записана одиниця, а до напруги $u_{0i}$ додасться $u_{0i+1} = u_{0i} / 2$, і в наступному також буде вироблено

порівняння $u_x$ і $u_{0i} + u_{0i}/2$ Цей процес триває до тих пір, поки не буде підібрано напруга, найбільш близьке до вхідного. Двійковий код потім перетворюється в десятковий і в цьому виді використовується в наступних блоках вольтметра. За таким принципом працюють також інші датчики.

АЦП з порозрядним врівноваженням і вольтметри на їх основі мають високу точність (погрішність 0,001%) і швидкодія (частота тактів більше 1 МГц).

При використанні аналогових датчиків з'являється потреба в перетворенні аналогової інформації в цифрову. Тому використання АЦП порозрядного врівноваження є доцільним для розширення функціональних можливостей системи.

### 2.6 Вибір типу диспетчеризації та алгоритму планування

В операційних системах реально часу існують два типи диспетчеризації: системи жорсткого та м'якого реально часу. Головна відмінність між ними полягає в реакції на подію. Для систем з жорстким реальним часом невчасне виконання поставленої задачі є критичним, ща в свою чергу призводить до відмов та неможливості виконання поставленого завдання. На практиці час реагування на будь-яку подію повинен бути мінімальним. Для системи м'якого реального часу існують відхилення від заданих параметрів, але на загальний стан системи такі відхилення не впливають. Також треба зазначити, що для таких систем існує необхідність зазначати терміни виконання для кожної задачі, при досяганні яких завдання повинно (для систем м'якого реального часу – бажано) бути виконано. Планувальник завдань використовує ці терміни для визначення пріоритету задачі при її запуску та виконанні[28].

Основна відмінність систем жорсткого і м'якого реального часу можна охарактеризувати так: система жорсткого реального часу ніколи не запізниться з реакцією на подію, система м'якого реального часу не повинна запізнюватися з реакцією на подію. Для виконання поставленої мети було

вирішено використовувати операційну систему м’якого реально часу, через відсутність жорстких вимог до часу виконання поставленої задачі.

Сучасні RTOS в своїй більшості є багатозадачними, тому в таких система виникає потреба у поділі обчислювальних ресурсів. За вирішення цієї проблеми відповідає планувальник задач, також він визначає послідовність та час, у який буде виконана задача.

Планувальник завдань — це модуль (програма), що відповідає за розподіл часу доступних процесорів між запущеними завданнями. Відповідає за перемикання завдань із стану блокування в стан готовності, а також за вибір завдання (завдань - за кількістю процесорів) з числа готових до виконання процесором. Існують наступні методи планування:

- Циклічний алгоритм (рис.2.12);
- FIFO-планування (рис 2.13);
- Кооперативна багатозадачність - тип багатозадачності, при якому наступне завдання виконується тільки після того, як поточне завдання явно оголосить себе готовою віддати процесорний час іншим завданням.;
- Пріоритетна багатозадачність з витісненням.

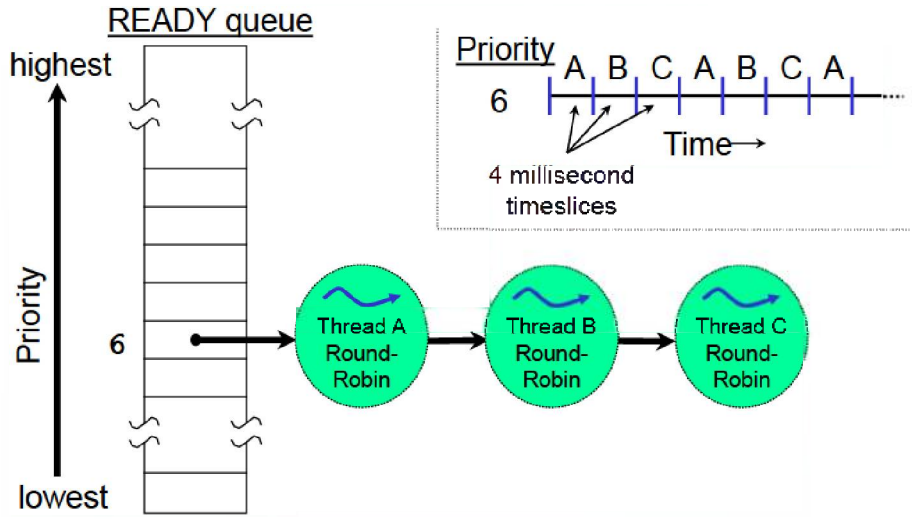



Рис. 2.12 – Схема роботи циклічного алгоритму

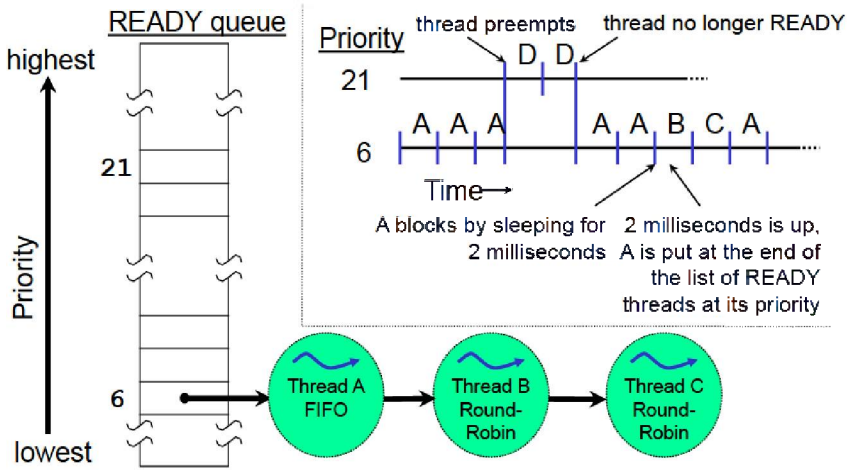



Рис. 2.13 – Схема роботи FIFO

Пріоритетна багатозадачність з витісненням є найбільш популярним методом в ОСРЧ. Задачі розподіляються за пріоритетами виконання, відповідно від висоти пріоритету залежить реактивність задачі. Саме підхід витісняючого планування забезпечує високу реактивність, шляхом витиснення низькопріоритетної задачі при появі задачі з більшим пріоритетом (рис. 2.14). Ураховуючи вищесказане, виникає розуміння того, що, при появі високопріоритетної задачі, усі ресурси будуть задіяні на її виконання, а задачі більш низького пріоритету будуть чекати поки вона не завершиться. Тому розробнику треба дуже ретельно ставитися до вибору пріоритетів для виконуваних задач. Кожен раз, коли планувальник завдань отримує сигнал про настання деякої зовнішньої події (тригеру) :

1) Визначає, чи повинна поточна виконувана завдання продовжувати працювати.
2) Встановлює, яке завдання має запускатися наступного.
3) Зберігає контекст зупиненої завдання (щоб вона потім відновила роботу з місця зупинки).
4) Встановлює контекст для наступного завдання.
5) Запускає цю задачу.

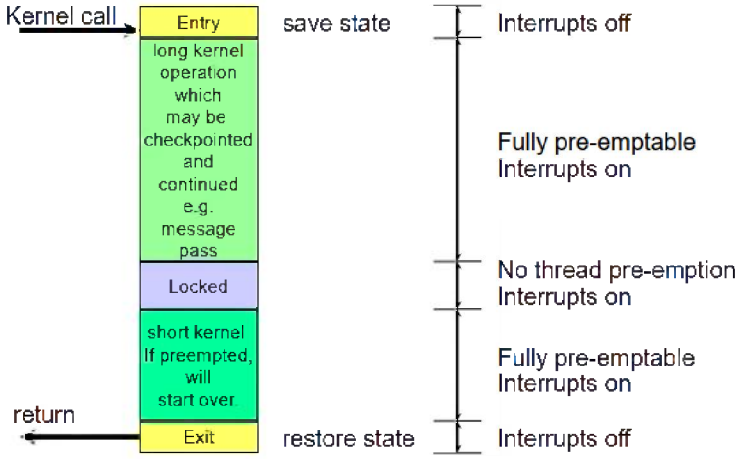



Рис. 2.14 – Схема роботи витисняючого планування

Також безпосередньо в RTOS різні методи планування можуть комбінуватися між собою, змінювати один одного в процесі роботи, змінювати пріоритетність задач, але головною вимогою для усіх методів планування залишається відповідність критеріям оптимального функціювання системи. Для систем з жорстким реальним часом такий параметр відомий, але для систем м'якого реального часу такі параметри треба визначати, це може бути мінімальне максимальне запізнення або середньозважена своєчасність завершення операцій.

Орієнтуючись на усе вищесказане, можна зробити висновки, що для якісного планування задач пріоритетними алгоритмами планування є витісняюче планування на основі пріоритетів з точками витіснення та витіснююче планування на основі пріоритетів з негайним витісненням. В першому алгоритмі існують рівновіддалені точки витіснення, при досягнені яких починає виконуватися готова до виконання задача з більшим пріоритетом, якщо така є в черзі. В другому алгоритмі планування реакція на переривання майже миттєва, за умови що в цей час не виконується код критичного розділу, який не можна переривати.

# РОЗДІЛ 3
# РОЗРОБКА АПАРАТНИХ КОМПОНЕНТІВ ТА ЇХ НАЛАШТУВАНЬ ДО СИСТЕМИ

## 3.1 Центральний пристрій системи.

IoT системи використовують різноманітні центральні пристрої для обробки, аналізу даних та створення на їх основі керуючих команд. У таблиці 3.1 представлені пристрої, які використовують як основополагаючі елементи в системах дистанційного керування.

***Таблиця 3.1***

**Характеристики пристроїв для IoT**

| Parameters | Arduino Uno | Arduino Yun | Intel Galileo Gen 2 | Intel Edison | Beagle Bone Black | Electric Imp 003 | Raspberry Pi B+ | ARM mbed NXP LPC1768 |
|---|---|---|---|---|---|---|---|---|
| Processor | ATMega328P | ATmega32u4, and Atheros AR9331 | Intel® Quark™ SoC X1000 | Intel® Quark™ SoC X1000 | Sitara AM3358BZCZ100 | ARM Cortex M4F | Broadcom BCM2835 SoC based ARM11 76JZF | ARM Cortex M3 |
| GPU | - | - | - | - | PowerVR SGX530 @520 MHz | - | VideoCore IV® Multimedia@ 250 MHz | - |
| Operating voltage | 5V | 5V, 3V | 5V | 3.3V | 3.3V | 3.3V | 5V | 5V |
| Clock speed (MHz) | 16 | 16,400 | 400 | 100 | 1 GHz | 320 | 700 | 96 |
| Bus width (bits) | 8 | 8 | 32 | 32 | 32 | 32 | 32 | 32 |
| System memory | 2kB | 2.5 kB, 64 MB | 256 MB | 1 GB | 512 MB | 120 KB | 512 MB | 32 KB |
| Flash memory | 32 kB | 32kB, 16 MB | 8 MB | 4 GB | 4 GB | 4 Mb | - | 512 KB |
| EEPROM | 1 kB | 1 kB | 8 kB | - | - | - | - | - |
| communication supported | IEEE 802.11 b/g/n, IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial | IEEE 802.11 b/g/n,IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial | IEEE 802.11 b/g/n,IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial | IEEE 802.11 b/g/n,IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial | IEEE 802.11 b/g/n, 433RF, IEEE 802.15.4, BLE 4.0, Ethernet, Serial | IEEE 802.11 b/g/n,IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial | IEEE 802.11 b/g/n,IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial | IEEE 802.11 b/g/n,IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial |
| Development environments | Arduino IDE | Arduino IDE | ArduinoIDE | Arduino IDE, Eclipse, Intel XDK | Debian, Android, Ubuntu, Cloud9 IDE | Electric Imp IDE | NOOBS | C/C++ SDK, Online Compiler |
| Programming language | Wiring | Wiring | Wiring, Wyliodrin | Wiring, C, C++, Node.JS, HTML5 | C, C++, Python, Perl, Ruby, Java, Node.js | Squirrel | Python, C, C++, Java, Scratch, Ruby | C, C++ |
| I/O Connectivity | SPI, I2C, UART, GPIO | SPI, I2C, UART, GPIO | SPI, I2C, UART, GPIO | SPI, I2C, UART, I2S, GPIO | SPI, UART, I2C, McASP, GPIO | SPI, I2C, UART, GPIO | SPI, DSI, UART, SDIO, CSI, GPIO | SPI, I2C, CAN, GPIO |

Проаналізувавши існуючі апаратні рішення по створенню IoT систем, було прийняте рішення, що головним апаратним компонентом системи буде одноплатний комп'ютер (ОК)[17]. Таке рішення пов'язано с тим, що ОК має необхідний набір параметрів таких, як: мережевий інтерфейс, процесор з досить великим обчислювальним потенціалом, підтримка мережевої операційної системи, наявність різноманітних фізичних інтерфейсів, малі розміри та низьку ціну. Серед ОК представлених на ринку було обрана Orange Pi PC H3 через наступний набір характеристик:

- процесор: H3 чотирьохядерний процесор Cortex-A7 H.265 / HEVC 4 K;

- GPU: Mali400MP2 GPU @ 600 мГц. Підтримує OpenGL ES 2.0;

- оперативна пам'ять (SDRAM): 1 ГБ DDR3 (спільно з GPU);

- пам'ять програм (MMC): micro-SD (макс. 64 ГБ) / MMC;

- мережевий інтерфейс: 10/100 Ethernet RJ45

- відео вхід: B CSI вхідний роз'єм камери;

- підтримує 8-бітові YUV422 CMOS Датчик Інтерфейс;

- підтримує CCIR656 протоколу для NTSC і PAL;

- підтримує відео захоплення 1080P @ 30fps;

- аудіо вхід: мікрофон;

- відео виходи: підтримка HDMI вихід з підтримкою HDCP;

- аудіо вихід: 3.5 мм роз'єм і HDMI;

- джерело живлення: вхід постійного струму;

- USB 2.0 порти: Три USB 2.0 Host, USB 2.0 OTG;

- порти введення / виводу: 40-контактний роз'єм, сумісний з Raspberry Pi B +

- підтримувані ОС: Android lubuntu, debian

- розміри плати: 85 х 55 мм.

Такий набір властивостей дозволить ОК виконувати задачі збору цифрових даних, їх обробку й аналіз, створення керуючих команд на основі отриманого аналізу та вподобань користувача, а також комунікацію з мережею Інтернет.

### 3.2 Розширення можливостей ОК.

З митою розширення можливостей ОК, до системи додається пристрій на базі мікропроцесору (МК). Це необхідно для повноцінної роботи з аналоговою інформацією, перетворення її на цифрові дані, а також обробки керуючих команд та застосування їх для кінцевих пристроїв (реле, двигун та ін.). Щоб реалізувати увесь потенціал МК було вибрано платформу Arduino, а саме Arduino Mega 2560. Ця платформа має безліч готових реалізацій для вирішення великої кількості задач, відома широкому колу користувачів і розробників та доволі проста в освоєнні. Arduino Mega 2560 серед усіх пристроїв платформи має найбільшу кількість фізичних інтерфейсів, тому найкраще задовольняє вимоги з підключення датчиків, мікроконтролерів, кінцевих пристроїв та ін..

Для отримання інформації про навколишнє середовище необхідно використовувати датчики, переважно цифрові для підвищення швидкодії системи та зменшення похибки, а також зменшення собівартості системи за рахунок використання цифрових шин даних. Такий принцип бажаний і для інших пристроїв системи (мікроконтролерів, цифрових реле, та інших девайсів з підтримкою цифрового інтерфейсу). У випадках де використання цифрових засобів не можливо, використовуються аналогові засоби.

Орієнтуючись на базовий принцип, у системі використовуються цифрові датчики температури та вологості DHT12, а також датчик атмосферного тиску

BMP280 та люксметр BH1750 (датчик освітленості). Це початковий набір пристроїв необхідний для взаємодії з навколишнім середовищем.

Також для впливу на навколишнє середовище необхідно впровадити силові електричні реле, які будуть відповідальні за вмикання/вимикання кінцевих пристроїв (системи вентиляції, кондиціювання, опалення, поливу та ін.). Керування реле відбувається через Arduino, який подає керуючий сигнал відповідного рівня на керуючий контакт реле (рис.3.2.).

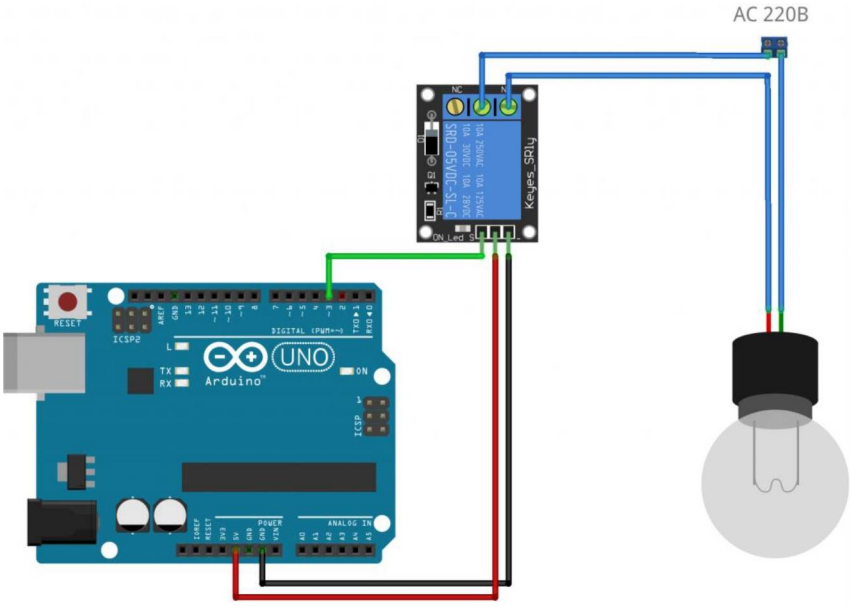



Рис.3.1 – Керування силовим реле через Arduino.

Отже представлений набір приладів надасть можливість ОК для моніторингу та керування системою, а також надасть гнучкості при вирішенні задач.

### 3.3 Розробка механізму між вузлової комунікації.

Використовуючи цифрові пристрої ми маємо можливість використання цифрових засобів комунікації. Для цього необхідно запровадити інтерфейси цифрових шин даних SPI та $I^2C$. Ці два інтерфейси мають свої переваги для створення обміну даними, які зазначені в таблиці 3.2 [18].

***Таблиця 3.2***

**Переваги інтерфейсів обміну даними**

| Переваги шини SPI | Переваги шини $I^2C$ |
|---|---|
| Простота протоколу передачі на фізичному рівні забезпечує високу надійність і швидкодію передачі. Гранична швидкодія шини SPI вимірюється десятками мегагерц і, тому, вона ідеальна для потокової передачі великих обсягів даних і широко використовується в високошвидкісних ЦАП / АЦП, драйвери світлодіодних дисплеїв і мікросхемах пам'яті | Шина I2C залишається двохдротовою, незалежно від кількості підключеної до неї мікросхем. |
| Всі лінії шини SPI є односпрямованим, що істотно спрощує рішення задачі перетворення рівнів і гальванічної ізоляції мікросхем | Можливість мультімастерной роботи, коли до шини підключено кілька провідних мікросхем. |
| Простота програмної реалізації протоколу SPI. | Протокол I2C є більш стандартизованим, тому, користувач I2C-мікросхем більш захищений від проблем несумісності обраних компонентів. |

I2C шина є однією з модифікацій послідовних протоколів обміну даних. У стандартному режимі забезпечується передача послідовних 8-бітних даних зі швидкістю до 100 кбіт / с, і до 400 кбіт / с в "швидкому" режимі. Для здійснення процесу обміну інформацією по I2C шині, використовується всього два сигнали лінія даних SDA лінія синхронізації SCL Для забезпечення реалізації двобічної шини без застосування складних арбітрів шини вихідні

каскади пристроїв, підключених до шини, мають відкритий стік або відкритий колектор для забезпечення функції монтажного "I" [19].

Проста дводротова послідовна шина I2C мінімізує кількість сполуки між пристроями, пристрої мають менше контактів і потрібно менше доріжок. Як результат - друковані плати стають простішими і технологічними при виготовленні. Інтегрований I2C-протокол усуває необхідність в дешифраторі адреси та іншої зовнішньої логіці узгодження.

Максимальна допустима кількість пристроїв, приєднаних до однієї шині, обмежується максимальним обсягом до шини 400 пФ. Вбудований в мікросхеми апаратний алгоритм завадостійкість забезпечує цілісність даних в умовах перешкод значної величини. Усі $I^2C$-сумісні пристрої мають інтерфейс, який дозволяє їм зв'язуватися один з одним по шині навіть у тому випадку, якщо їх напруга живлення істотно відрізняється.

Ознайомившись з перевагами шини даних $I^2C$, прийнято рішення про використання цього інтерфейсу для комунікації цифрових пристроїв (датчиків та мікроконтролерів) на одній дводротовій шині даних, це дозволить реалізувати можливість діагностики системи шляхом опитування вузлів, полегшить розгортання системи, знизить її собівартість та підвищить якість керування.

Шину SPI буде доречним використовувати для зв’язку між ОК та МП, бо у цьому випадку необхідно досягти більшої швидкодії для коректного керування та слідкування за станом системи. Також цей інтерфейс можна використати для створення бездротового зв’язку використавши модуль зв’язку nRF24L01+ , який також підтримується Arduino Mega 2560 та Orange Pi PC H3. Для повного розуміння взаємодії компонентів в системі дистанційного керування необхідно розглянути відповідну схему (рис.3.2.) .

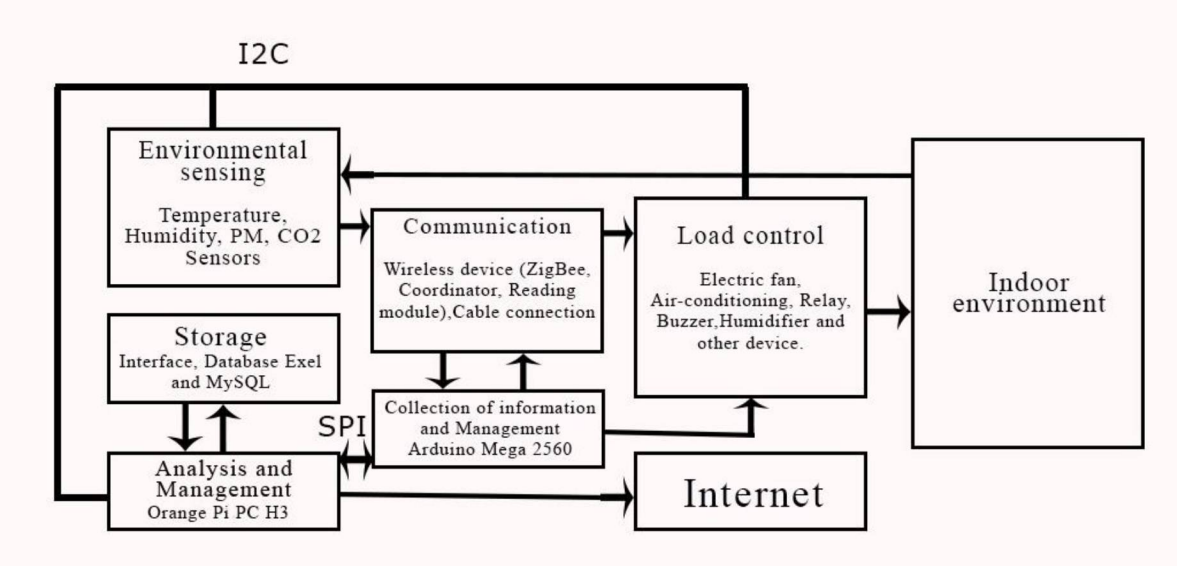



Рис.3.2 – Схема взаємодії компонентів в системі дистанційного керування.

Для відповідності стандартам IoT система повинна мати можливість планування та прогнозування. Щоб забезпечити цей функціонал необхідно зберігати оброблені дані, щоб на їх основі отримати знання та вибірку даних для навчання системи, а також для налагодження адекватних алгоритмів роботи. Збереження даних може забезпечити база даних MySQL підключена до ОК, а у випадках відсутності Інтернету, фізичний носій під'єднаний також до ОК.

Запропонована реалізація повністю виконує вимоги моделі та може бути реалізована в працездатну систему.

# РОЗДІЛ 4
# РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 4.1 Вибір та впровадження операційної системи реально часу.

Для вирішення поставленої задачі було обрано операційну систему реального часу від компанії QNX, бо це стабільний комерційний продукт, який дає розробникам та науковцям ліцензію для реалізації нових проектів.

Основним призначенням операційної системи QNX є реалізація програмного інтерфейсу POSIX у масштабованій, відмовостійкій формі, що підходить для широкого кола відкритих систем, починаючи від невеликих вбудованих систем з обмеженими ресурсами і закінчуючи великими розподіленими обчислювальними середовищами.[24]

Відмовостійка архітектура також вкрай необхідна для роботи критично важливих додатків, у ОС QNX повною мірою використовуються можливості роботи з MMU-обладнанням.

Система досить невелика, щоб у мінімальній комплектації поміститися на одну дискету, разом з цим вона вважається дуже швидкою і належним чином «закінченою» (яка практично не містить помилок).

QNX ідеально підходить для вбудовуваних програм реального часу. Вона може бути масштабована до компактних конфігурацій і здатна працювати в багатозадачному режимі, управляти потоками, здійснювати планування процесів за пріоритетами і виконувати швидке перемикання контекстів. Більш того, операційна система надає всі ці можливості за допомогою програмного інтерфейсу, що базується на стандартах POSIX. Таким чином, компактність системи досягається не на шкоду стандартам.

Як мікроядерна операційна система, QNX заснована на ідеї роботи основної частини своїх компонентів як невеликих завдань, званих сервісами. Це відрізняє її від традиційних монолітних ядер, у яких ядро операційної

системи — одна велика програма, що складається з великої кількості «частин», кожна зі своїми особливостями. Це дозволяє ОС QNX володіти досить великою гнучкістю. Користувачі (розробники) можуть легко змінювати її конфігурацію відповідно до вимог створюваних додатків. Можна використовувати лише ті ресурси, які необхідні для конкретного завдання, змінюючи систему в діапазоні від мінімальної конфігурації мікроядра з декількома невеликими модулями до повнофункціональної системи, призначеної для обслуговування сотень користувачів (рис.4.1).

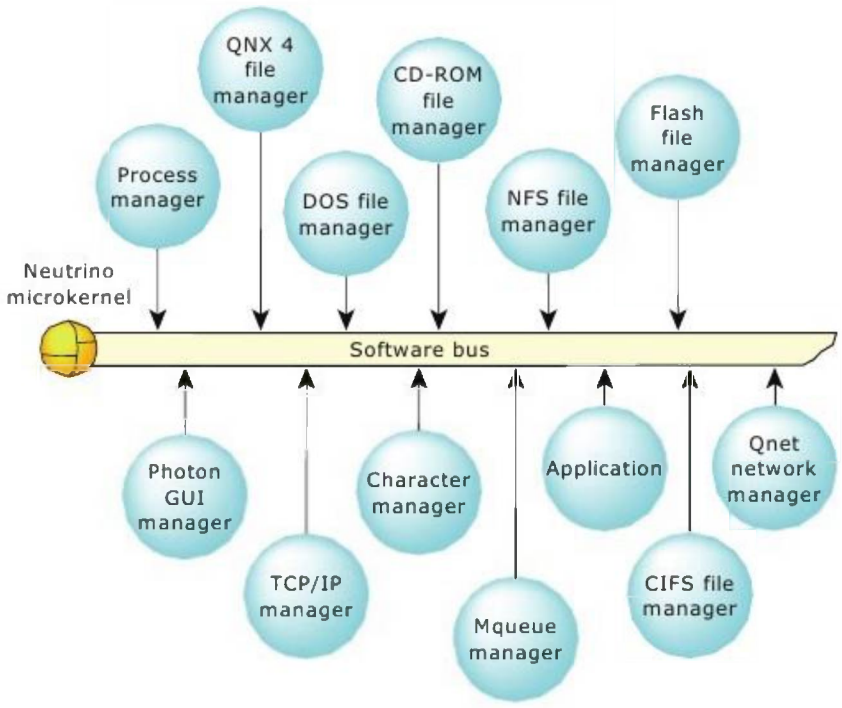



Рис.4.1 – Архітектура ОС QNX

QNX Neutrino, випущена в 2001 році, перенесена на багато платформ і зараз здатна працювати практично на будь-якому сучасному процесорі, що використовується на ринку систем, що вбудовуються. Серед цих платформ присутні сімейства X86, MIPS, PowerPC, а також спеціалізовані сімейства процесорів, такі як SH-4, ARM, StrongARM та XScale.

Обравши операційну систему реального часу QNX, виника потреба портування її до arm64 платформи.

Портування - адаптація деякої програми або її частини, з тим щоб вона працювала в іншому середовищі, що відрізняється від того середовища, під

яку вона була спочатку написана з максимальним збереженням її користувацьких властивостей.

Необхідність у виконанні портування виникає зазвичай через відмінності в системі команд процесора, відмінностей між способами взаємодії операційної системи і програм (API - Application Program Interface), принципових відмінностей в архітектурі обчислювальних систем, або через деяких несумісностей або навіть повної відсутності мови програмування в цільовому оточенні.

Міжнародні стандарти (зокрема, що просуваються ISO) значно спрощують портування [26], завдяки тому що вони описують середовище виконання програм таким чином, що відмінності між платформами стають мінімальними. Часто портування програм між платформами, які реалізують один і той же стандарт (такий як POSIX.1) зводяться до перекомпіляції програми на новій платформі.

Існує також все розширюваний набір інструментів, що полегшують портування, наприклад, таких як GCC, що надає незмінну мову програмування на різних платформах.

Деякі мови програмування високого рівня досягають портування шляхом трансляції вихідного коду в проміжну мову, що має компілятори для багатьох процесорів і операційних систем.

Портування на архітектуру arm64(найпоширенішу в системах IoT) реалізовано шляхом перебудови та компіляції ядра ОС з подальшим розгортанням системи на макетній платі.

### 4.2 Розробка ПЗ для комунікації.

Щоб спілкуватися з будь-якими зовнішніми пристроями і керувати ними, Orange Pi має на борту інтерфейс GPIO (англ. General-purpose input / output). GPIO це низькорівневий інтерфейс вводу / виводу загального призначення створений для прямого управління. Тобто через цей інтерфейс Orange може слухати і віддавати команди будь-якому зовнішнього пристрою.

Для роботи з GPIO необхідно встановити бібліотеки WiringPi або WiringOP на Orange.

WiringPi (WiringOP для Orange) [20] написана на мові C, випущена під ліцензією GNU LGPLv3 і придатна для використання в C, C ++ і RTB (BASIC), а також в інших мовах (але для цього потрібні спеціальні функції-обгортки). Бібліотека WiringPi створювалася з прицілом на схожість з мовою Wiring, який використовується в Arduino. Оригінальна бібліотека дає можливість роботи з 26-контактним GPIO-коннектором, який підтримує різні інтерфейси і інші формати передачі даних. Серед них були 8 цифрових GPIO-контактів - їх можна було запрограмувати і на цифровий вхід, і на цифровий вихід. Крім того, два з них можна було виділити для видачі апаратної ШІМ. До того ж у них був 2-провідний інтерфейс $I^2C$, 4-провідний інтерфейс SPI (плюс ще один контакт для SS, тому в цілому виходить 5 контактів) і послідовний інтерфейс UART з 2 додатковими контактами. Реалізацію інтерфейсів $I^2C$ та SPI наведено у Додатку В, Г та Д.

Для забезпечення бездротового зв'язку між ОК та МП необхідно підключити радіомодуль NRF4L01 до обох пристроїв та написати відповідне ПЗ для їх роботи. NRF4L01 працює з допомогою SPI шини даних.

На Orange Pi обмін даними зарез NRF4L01 реалізований наступним чином. Радіомодуль приєднується до відповідних контактів SPI шини. Далі створюємо код для обміну даними. В першу чергу підключаємо бібліотеки для роботи з NRF4L01: `#include <RF24/nRF24L01.h>`, `#include <RF24/RF24.h>`. Після цього створюємо масив для отримання даних: `uint8_t saveData[MAX_LEN]; uint8_t pinNumber; uint8_t loadSize;`. Приймачу можна задати до 6 труб(прослуховувачів) функцією `openReadingPipe (номер, адресу)`. З номерами труб від 0 до 5 і адресами труб збігаються з адресами труб передавачів. Далі треба визначити номери прослуховуючи труб: `uint8_t pinAddress[адреса][адреса] = {'0'(номер),'адреса передавача'}`. Наступним кроком

створюємо об'єкт radiolisn для роботи з бібліотекою RF24 та вказуємо номери виведення CE і SPI порту: `RF24 radiolisn(BCM_PIN, SPI_DEV);`. Далі проводимо налаштування приймача: ініціалізуємо, обираємо режим роботи, час очікування, дозволяємо динамічну зміну блоку даних для усіх труб, відкриваємо труби для прийому даних, задаємо канал передачі даних, зазначаємо потужність передавача, його швидкість передачі даних. Зробивши це вмикаємо приймач та починаємо прослуховування: `radiolisn.startListening(); radiolisn.printDetails();`. Прослуховування відбувається у безкінечному циклі та у разі отримання даних, читаємо їх до масиву та зазначаємо скільки байт читати.

Для передачі даних з Arduino необхідно приєднати NRF4L01 до плати. Після чого в скетчі зробити наступні дії. В першу чергу підключаємо бібліотеки для роботи з NRF4L01: `#include <RF24/nRF24L01.h>, #include <RF24/RF24.h>`. Наступним кроком створюємо об'єкт radiosent для роботи з бібліотекою RF24 та вказуємо номери виведення CE і SPI порту: `RF24 radiosent(BCM_PIN, SPI_DEV);`.Після цього створюємо масив для відправки даних: uint8_t sentData[розмір]; uint8_t loadSize; та визначаємо номери труб передавача: `uint8_t pinAddress[адреса][адреса] = {'0'(номер),'адреса передавача'}`. Далі проводимо налаштування приймача: ініціалізуємо, обираємо режим роботи, дозволяємо динамічну зміну блоку даних для усіх труб, відкриваємо трубу для передачі даних, зупиняємо прослуховування радіоефіру, задаємо канал передачі даних, зазначаємо потужність передавача, його швидкість передачі даних. Зробивши налаштування у методі loop надсилаємо дані: `for (uint8_t i = 0; i < payloadSize; i++) { Serial.print(" "); Serial.print((int) sentDate[i]);`. Після чого робимо затримку 1000мс.

В інтерфейс $I^2C$ дані передаються по двох дротах - дріт даних і дріт тактів синхронізації. У мережі є хоча б одне провідний пристрій (Master), яке

ініціалізує передачу даних і генерує сигнали синхронізації. У мережі також є ведені пристрої (Slave), такі як гіроскопи, акселерометри, датчики тиску, EEPROM-пам'ять, дисплеї, які передають дані по запиту ведучого. У кожного відомого пристрою є унікальний адресу, за якою ведучий і звертається до нього. Адреса пристрою вказується в паспорті (datasheet). До однієї шині I2C може бути підключено до 127 пристроїв, в тому числі кілька ведучих. До шині можна підключати пристрої в процесі роботи, тобто вона підтримує «гаряче підключення».

Pi4J надає можливість роботи з послідовною шиною $I^2C$ з Java на Orange Pi. Всі класи і інтерфейси для ініціалізації і роботи з шиною знаходяться в пакеті com.pi4j.io.i2c. *;. Для початку роботи з $I^2C$ підключаємо бібліотеку Pi4J. Наступним кроком підключається пристрій, який має свою адресу. Адреса датчику температури та вологості в шістнадцятирічній системі дорівнює 0xB8. Оголошуємо константу: `private static final int DHT12_ADDRES = 0xB8;`. Так як дані будуть записані тільки в регістри температури та вологості (від 0x00 до 0x04), оголошуємо тільки їх: `private static final int DHT12_HUMID_ADDRES = 0x00; private static final int DHT12_HUMID_FRACTION_ADDRES = 0x01; private static final int DHT12_TEMPER_ADDRES = 0x02; private static final int DHT12_TEMPER_FRACTION_ADDRES = 0x03; private static final int DHT12_CONTROL_ADDRES = 0x04;`. Далі потрібно явно зазначити платформу, для цієї системи - це Orange Pi: `PlatformManager.setPlatform(Platform.ORANGEPI);`. Щоб працювати з I2C пристроями, треба створити екземпляр I2CBus за допомогою методу getInstance класу I2CFactory, де параметр - це номер шини : `I2CBus i2cDateBus = I2CFactory.getInstance(I2CBus.BUS_0);`. Створюємо екземпляр I2CDevice за допомогою методу getDevice, де параметр - це адреса пристрою. Таким чином можна створити об'єкт для кожного підключеного пристрою: `I2CDevice i2cDevice =`

`i2cDateBus.getDevice(DHT12_ADDRES);`.Після цього маємо можливість прочитати дані за допомогою функції read(бажане значення). Таким чином опитуємо кожен пристрій на шині даних $I^2C$.

Представлений набір цифрових інтерфейсів має великий потенціал для розширення та забезпечує надійний зв'язок.

### 4.3 Розробка TCP/IP серверу для окремих підсистем.

Для дистанційного керування системою необхідно створити TCP/IP сервер. За допомогою IP-адреса з'являється можливість однозначної ідентифікації в мережі, а номер порту дозволяє ідентифікувати конкретне з'єднання між різними пристроями. Завдяки цьому можна здійснити дистанційне підключення та керування [21].

При побудові класичного TCP серверу можна виділити три складові частини: демонізація процесу, робота з сокетами, створення дочірнього процесу і обробка його завершення. При роботі з сокетами необхідно: створити дескриптор сокетів, призначити сокету адресу, прийняти запит на установку з'єднання, очикування вхідного зєднання, прийняти / відправити дані. Лістинг програми наведено у Додатку Е.

Дескриптор сокетів:

```
id2=2; //2-я переменная, для ID потока 2
      result = pthread_create(&thread2, NULL, thrd_2, &id2);
      if (result==0) print_mes("thread TCP create\n",1,0); else
            print_mes("thread TCP create error\n",1,1);
```

Призначення сокету адресу:

```
//Bind
      if(  bind(socket_desc,(struct  sockaddr  *)&server  ,
sizeof(server)) < 0)
      {
            //print the error message
            print_mes("bind TCP failed. Error\n",1,1);
            pthread_exit(NULL); //завеершаю поток
```

```
        }
        puts("bind done\n");
```

Прийняття запиту на з'єднання:

```
//Listen
        listen(socket_desc , 3);
```

Очкування вхідного з'єднання:

```
        //Accept and incoming connection
        puts("Waiting for incoming connections...\n");
        c = sizeof(struct sockaddr_in);
        pthread_t thread_id;
        while(  (client_sock  =  accept(socket_desc,  (struct
sockaddr *)&client, (socklen_t*)&c)) )
        {
            puts("Connection accepted\n");
            //создаю новые потоки для каждого клиента
            if(  pthread_create(  &thread_id  ,  NULL  ,
connection_handler , (void*) &client_sock) < 0)
            {
                print_mes("could not create thread\n",1,1);
                pthread_exit(NULL); //завеершаю поток
            }
            puts("Handler assigned\n");
        }
        if (client_sock < 0)
        {
            perror("accept failed");
            pthread_exit(NULL); //завеершаю поток
        }
```

Прийняття / відправка даних:

```
//Receive a message from client
        while( (read_size = recv(sock , client_message , BUF_SIZE
, 0)) > 0 )
        {
         if (read_size <= 2048) decodeTcpPacket(client_message,
sock);              //обработка пакета
```

```
            memset(client_message, 0, BUF_SIZE);
        }
```

Отже створений TCP/IP сервер забезпечує можливість дистанційного моніторингу та керування, що є однією з основних вимог до системи.

### 4.3 Створення веб-додатку

Після створення TCP серверу з'явилася можливість дистанційного доступу до системи, але для взаємодії з користувачем необхідно створити комфортний графічний інтерфейс. Найкращім рішенням буде створення веб-додатку в браузері. Це дозволить отримувати доступ з будь-якого гаджету з доступом до Інтернету.

Для створення функціонального інтерфейсу користувача використовуються наступні засоби розробки JavaScript (реалізація скрипту), фреймворк CppCMS (створення структури веб-додатку),CSS (створення зовнішнього вигляду) та HTML (розмітка гіпертексту )[22].

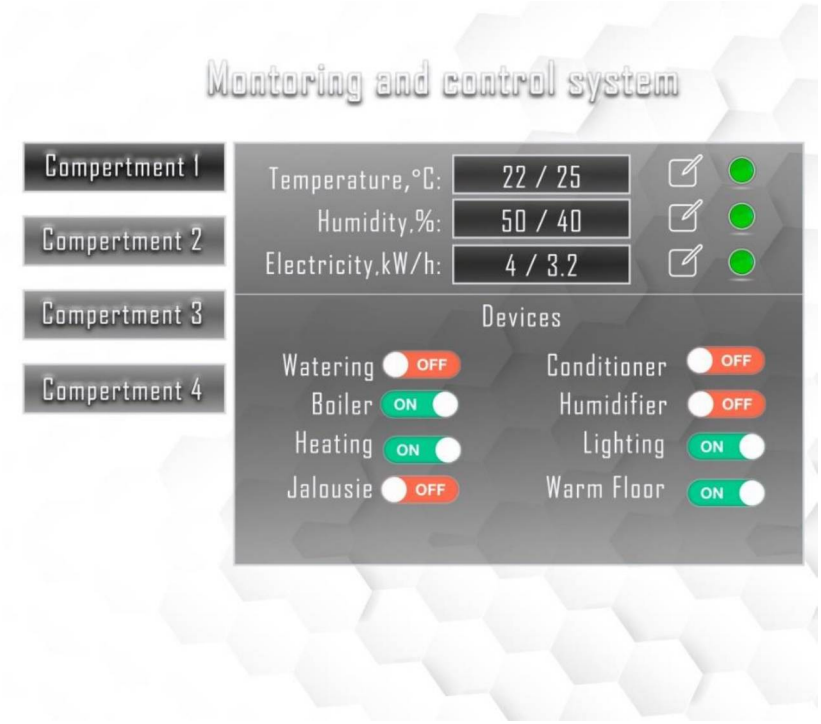



Рис.4.1 – Графічний інтерфейс користувача.

Завдяки цим засобам вдалося створити веб-додаток у вигляді веб-сторінки (рис.4.1.), яка дозволяє слідкувати за показниками та у разі чого відразу корегувати їх, а за допомогою реле є можливість дистанційного керування пристроями. Також є можливість роботи за створеним сценарієм, це дає можливість без зайвих дій налаштувати систему, але на даний час робота сценаріїв не дуже адекватна, через нестачу вибірки даних та потребує налагодження для безперебійної роботи.

Створений інтерфейс є кінцевим етапом у створенні дистанційної системи моніторингу та керування, бо усі вимоги до системи виконані.

# РОЗДІЛ 5. ЛАБОРАТОРНІ ВИПРОБУВАННЯ СИСТЕМИ ТА ЇЇ МОДУЛІВ

Для перевірки дієздатності розробленої системи дистанційного моніторингу та управління виконувалися лабораторні дослідження. Випробовувалися керуючі та апаратно-програмні складові системи, TPC/IP сервер, а також інтерфейс $I^2C$ та датчики.

Тестування проводилося з використанням сертифікованого обладнання та довело відповідність системи потрібним вимогам. Застосовувалося як ручне, так і автоматичне керування пристроями. Максимальні похибки показників не перевищували двох відсотків, що символізує працездатність системи.

Перевірка апаратно-програмної складової відбувалася за допомогою веб-додатку. Оператор спочатку змінював показники та технічні характеристики системи, а потім вимірював та контролював їх. Корегувалися температурні параметри, рівні вологості, енергоспоживання та освітлення, відбувалося вмикання та вимикання пристроїв. Результати випробувань та відповідність показників наведені в додатку Б.

Керуюча складова теж вдало впоралася з випробуваннями. Для її перевірки з одноплатного комп'ютера на мікропроцесор подавалися різні сигнали. Спочатку відправляли команду про вмикання реле, а через певний час сигнал про вимкнення. В ході тестування системи зв'язку та обробки команд ніяких проблем не виявлено.

Інтерфейси $I^2C$ та датчики випробувалися з використанням сторонніх вимірювальних пристроїв через порівняння їх показників. Адресне опитування пристроїв довело, що данні датчиків мають лише незначні відхилення від потрібних показників.

За допомогою програмного емулятора Hercules перевірявся TCP/IP – сервер. Його робота відповідає поставленим задачам. Результати випробувань дають можливість вважати, що мета кваліфікаційної роботи досягнута.

# ВИСНОВКИ

У цій кваліфікаційній роботі розроблено інтегровану InternetofThings (IoT) систему, яка оцінює та аналізує вузол для вимірювання параметрів навколишнього середовища в приміщенні та правила нечіткого контролю згідно експериментальних даних та проводить опитування для узагальнення роботи.

Для досягнення оптимальних умов в приміщеннях, які дають можливість проводити необхідну діяльність, важлива увага в роботі приділяється саме інтегрованості InternetofThings-системи, здатності підключення багатьох датчиків та якісного контролю навантажувальних пристроїв.

Щоб визначити надійний та стабільний вузол для вимірювання, розроблена система виконує аналіз навколишнього середовища за допомогою алгоритму. Використовується метод мінімальної варіації та мінімального середнього відхилення. Стабільність системи залежить від величини відхилення даних, чим більш вони м'які, тим більш ефективний та помітний контроль. Та навпаки, стабілізувати систему дуже важко при надмірно великих відхиленнях даних вузла для вимірювання.

Використання веб-додатку при керуванні системою забезпечує комфорт користувача. Розроблена система швидка, гнучка та здатна використовувати засоби IoT завдяки архітектурі одноплатного комп'ютера, який підключений з використанням бездротового зв'язку. Він об'єднує найкращі властивості мікропроцесора та комп'ютера. За допомогою мікропроцесора на архітектурі Arduino виконується керування силовими пристроями та збирання й первинна обробка аналогових даних.

Завдяки використанню існуючих апаратних рішень вдалося скоротити витрати на реалізацію системи. Це було важливою умовою, щоб зробити її конкурентоспроможною. Система доступна, стабільна та повністю відповідає вимогам IoT.

Відсутність помилок та збоїв при тестуванні системи дистанційного моніторингу та керування символізує її ефективність та придатність для прикладного використання. При необхідності вона може бути адаптована для промислової або побутової сфери.

Для допомоги розробникам та спрощення розгортання системи, планується створення платформи для підтримки елементної бази. В майбутньому буде необхідно лише приєднати пристрій із зазначенням його інтерфейсу та виду.

Одержані результати досліджень дають підставу вважати, що всі цілі кваліфікаційної роботи досягнуті, а задачі реалізовані. В підсумку була створена дієздатна та загальнодоступна система дистанційного моніторингу та керування, яка використовує апаратно-програмну платформу на архітектурі одноплатного комп'ютера та має реальну конкурентоспроможність.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. IoT и системы управления умным домом.[Електронний ресурс].режим доступа: URL: https://cyberleninka.ru/article/n/iot-i-sistemy-upravleniya-umnym-domom (дата звернення 11.05.2021)
2. Інтернет речей: проблеми правового регулювання та впровадження: Матеріали науково-практичної конференції. 24 жовтня 2017 р., м. Київ. / Упоряд. : В. М. Фурашев, С. Ю. Петряєв. – Київ : Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» Вид-во «Політехніка». 2017 – 238 с
3. Сферы применения IoT [Електронний ресурс]. режим доступа: URL: https://www.intel.ru/content/www/ru/ru/internet-of-things/industry-solutions.html (дата звернення 22.05.2021)
4. Плюсы и минусы технологии Интернета вещей. [Електронний ресурс]. режим доступа: URL: https://tsa.su/news/pljusy-i-minusy-tehnologii-interneta-veshhej/ (дата звернення 27.05.2021)
5. Социальные последствия развития Интернета вещей (IoT). [Електронний ресурс]. режим доступа: URL: https://cyberleninka.ru/article/n/sotsialnye-posledstviya-razvitiya-interneta-veschey-iot/viewer (дата звернення 29.05.2021)
6. 2020 рік стане переломним для Інтернету речей . [Електронний ресурс]. режим доступа: URL: https://www.itweek.ru/iot/article/detail.php?ID=210539 (дата звернення 01.09.2021)
7. 12 лучших систем «умный дом» .[Електронний ресурс]. режим доступа: URL: https://vyboroved.ru/reyting/luchshie-sistemy-umnyj-dom (дата звернення 08.09.2021)

8. Трехслойная концепция модель Интернета вещей. [Електронний ресурс]. режим доступа: URL:https://russianblogs.com/article/93732022258/ (дата звернення 12.09.2021)
9. Internet of Things: Architectures, Protocols, and Applications [Електронний ресурс]. режим доступа: URL: https://www.hindawi.com/journals/jece/2017/9324035/ (дата звернення 15.09.2021)
10. Кращі системи "Розумний будинок" [Електронний ресурс]. режим доступа: URL:https://vencon.ua/ua/articles/reyting-sistem-umnyy-dom-po-proizvoditelyam (дата звернення 19.09.2021)
11. 12 лучших систем «умный дом». [Електронний ресурс]. режим доступа: URL:https://vyboroved.ru/reyting/luchshie-sistemy-umnyj-dom (дата звернення 23.09.2021)
12. «IoT архитектура — первый взгляд под капот» 14.08.2018 [Електронний ресурс]. URL: https://habr.com/ru/post/420173/
13. «IoT архітектура» 10.07.2019 [Електронний ресурс]. URL: https://habr.com/ru/post/455377/
14. Блог Vencon. Рейтинг систем "Розумний будинок" по виробниках. [Електронний ресурс]. URL: https://vencon.ua/ua/articles/reyting-sistem-umnyy-dom-po-proizvoditelyam (дата звернення: 14.11.2019)
15. «7 IoT Business Models That Are Transforming Industries» 15.08.2018 [Електронний ресурс]. URL: https://danielelizalde.com/monetize-your-iot-product/
16. «Вычислительные машины, системы и сети» 15.02.2014 [Електронний ресурс]. URL: https://studfile.net/preview/717916/
17. Techopedia. Single-Board Computer (SBC). Дата оновлення: 25.01.2017. [Електронний ресурс]. URL: https://www.techopedia.com/definition/9266/single-board-computer-sbc
18. «Последовательный интерфейс SPI (3-wire)» 31.05.13 [Електронний ресурс]. URL: http://www.gaw.ru/html.cgi/txt/interface/spi/index.htm

19. «Интерфейсная шина IIC (I2C)» 16.04.2009 Електронний ресурс]. URL: http://easyelectronics.ru/interface-bus-iic-i2c.html
20. Офіційний сайт бібліотеки Wiring Pi. [Електронний ресурс]. URL: http://wiringpi.com/ (дата звернення: 21.02.2020)
21. «Веб-сервер на C++ и сокстах» 30.04.2015 [Електронний ресурс]. URL: https://code-live.ru/post/cpp-http-server-over-sockets/
22. Oliver J. HTML & CSS is hard. [Електронний ресурс]. URL: https://www.internetingishard.com/html-and-css/introduction/ (дата звернення: 16.03.2020)
23. What is RTOS and why you should use it for your IoT applications / Karim Hamdy, 2021. URL: https://www.zerynth.com/blog/what-is-rtos-and-why-you-should-use-it-for-your-iot-applications/ (дата звернення: 27.09.2021)
24. Операционная система реального времени QNX Neutrino 7.1. Системная архитектура / СВД Встраиваемые системы, 2020. URL: http://www.kpda.ru/upload/docs/Neutrino_Sys_Arch_6.5.pdf (дата обращения: 27.09.2021)
25. ОСРВ / Aics, 2017. – URL: http://fas.aics.ru/student/lectures/rts/owerview.pdf (дата звернення: 13.10.2021)
26. Операционные системы реального временя (ОСРВ)/ ПУЭ8. URL: http://puc8.ru/protsessory/699-operatsionnye-sistemy-realnogo-vremeni-osrv.html (дата обращения 7.10.2021)

# ДОДАТКИ

**Додаток А.**

## Завдання на кваліфікаційнуну роботу

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) Магістр
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 –Комп'ютерна інженерія

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри теоретичної та прикладної системотехніки**
**_____ д.т.н., проф. Шматков С. І.**

«____» _____ 20___ року

**З А В Д А Н Н Я**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Жинаги Владлена Владимировича

1. Тема роботи «Програмно-апаратна система для інтернету речей на ARM архітектурі»
керівник роботи Малахова М. О., кандидат технічних наук,

затверджені наказом по університету від "___" _____ 20__ року №_____

2. Строк подання студентом роботи_____

3. Перелік питань, які потрібно розробити)
- 3.1 Аналіз науково-технічної літератури та інформації в Інтернеті.
- 3.2 Розробка моделі та алгоритмів окремих підсистем.
- 3.3 Аналіз апаратних компонентів та їх налаштування до системи.
- 3.4 Розробка програмного забезпечення.
- 3.5 Лабораторні випробування системи та її модулів.

4. План роботи

| № з/п | Назви етапів роботи | Дата |
|---|---|---|
| 1 | Огляд науково-технічної літератури та джерел в Інтернеті | 15.10.20 – 01.02.21 |
| 2 | Розробка моделі системи | 02.02.21 – 19.03.21 |
| 3 | Розробка окремих підсистем | 20.03.21 – 09.04.21 |
| 4 | Аналіз апаратних елементів | 10.04.21 – 16.05.21 |
| 5 | Розробка програмного забезпечення | 24.05.21 – 30.06.21 |
| 6 | Лабораторні випробування системи та її модулів | 30.06.21 – 12.07.21 |
| 7 | Створення презентації | 06.07.21 – 19.08.21 |
| 8 | Підготовка доповіді на конференцію | 20.08.21 – 03.09.21 |
| 9 | Представлення дипломного проекту керівнику дипломної роботи та рецензенту | 21.09.21 – 05.10.21 |
| 10 | Розробка пояснювальної записки | 06.10.21 – 05.11.21 |
| 11 | Підготовка до захисту дипломної роботи | 06.11.21 – 21.11.21 |

5. Дата видачі завдання _____

Студент _____Живага В.В._____ _____ _____
(підпис) (прізвище)

Керівник роботи _____М. О. Малахова_____ _____ _____
(підпис) (прізвище)

Додаток Б

Затверджую

_____

«____»_____2021 р.

**Технічне завдання**
на розробку програмного виробу
«Система дистанційного моніторингу та керування»

| Назва розділу | Назва і зміст підрозділу |
|---|---|
| 1. Введення | 1.1. Назва виробу: система дистанційного моніторингу та керування.<br>1.2. Галузь застосування: побутова, медична та промислова. |
| 2. Підстава для розробки | 2.1. Навчальний план за спеціальністю 123 – «Комп'ютерна інженерія»<br>2.2. Завдання на кваліфікаційну роботу бакалавра затверджено наказом ректора № 0210-05/1804 від 10.09.2021. |
| 3. Призначення розробки | 3.1. Мета розробки програмного виробу: створення системи дистанційного моніторингу та управління на ARM архітектурі.<br>3.2. Призначення виробу: система дистанційного моніторингу та керування кіберфізичним експериментом значно збільшує можливості науковців та дослідників надаючи їм можливість користуватися науковим обладнанням незалежно від місця їх розташування (особливо при сучасній епідеміологічній ситуації в світі), а операційна система реального часу покликана зберегти обладнання при виникненні небезпеки нього.. |
| 4.Вихідні дані | У якості прототипу взяти модель описану у Kranz M. Building the Internet of Things: Implement New Business Models, Disrupt Competitors, Transform Your Industry. 2016. p. 272. В основу протоколу комунікації взяти Philips Semiconductors. The I2C-bus and how to use it. (including specifications). 1995 update. 3-3. April 1995. 1.0 THE I2C-BUS BENEFITS DESIGNERS AND. |
| 5. Технічні вимоги до виробу | 5.1. Вимоги до функціональних характеристик: можливість моніторингу навколишнього середовища та керування певними пристроями.<br>5.2. Вимоги до надійності: забезпечення працездатності системи при несправності другорядного модуля, забезпечення перешкодостійкого обміну даними.<br>5.3. Вимоги до умов експлуатації: встановлення програмно-апаратних засобів та живлення, під'єднання головного пристрою до локальної мережі, наявність браузера для керування системою.<br>5.4. Вимоги до складу і параметрів технічних засобів: пристрій з підтримкою підключення до мережі та наявністю браузера, локальна мережа.<br>5.5. Вимоги до інформаційної та програмної сумісності: підтримка з різними ОС та різними браузерами.<br>5.6. Вимоги до маркування та упаковки: жорстка упаковка з елементами пінопласту, маркування на українській та англійській мовах. |

<table>
<tr><td></td><td colspan="2">5.7. Вимоги до транспортування і зберігання: транспортування в упаковці будь-яким способом, температура транспортування/зберігання +10-+20 С°.<br>5.8. Спеціальні вимоги: відсутні.</td></tr>
<tr><td>6. Вимоги до документації.</td><td colspan="2">Документацією до виробу «Система віддаленого управління» вважати:<br>1) Справжнє Технічне завдання на розробку виробу (представити як Додаток Б до пояснювальної записки до кваліфікаційної роботи).<br>2) Програму і методику випробувань розробленого виробу (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи).<br>3) Опис виробу (представити у Розділі 3 до пояснювальної записки до кваліфікаційної роботи).<br>4) Фрагменти тексту програми (представити у Додатках до пояснювальної записки до кваліфікаційної роботи).</td></tr>
<tr><td>7. Техніко-економічні показники</td><td colspan="2">Орієнтовна оцінка ефективності виконуваної роботи: формування команд не більше 5 мкс, задача складності O($n^2$) для 1000 елементів не більше 100 мс; Аналіз аналогів представити в розділі 1.</td></tr>
<tr><td rowspan="8">8. Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td>15.10.20 – 01.02.21</td><td>1. Аналіз науково-технічної літератури та джерел в Інтернеті;</td></tr>
<tr><td>02.02.21 – 19.03.21</td><td>2. Розробка моделі системи;</td></tr>
<tr><td>20.03.21 – 09.04.21</td><td>3. Розробка окремих підсистем;</td></tr>
<tr><td>10.04.21 – 16.05.21</td><td>4. Аналіз апаратних елементів;</td></tr>
<tr><td>24.05.21 – 30.06.21</td><td>5. Розробка програмного забезпечення;</td></tr>
<tr><td>30.06.21 – 12.07.21</td><td>6. Лабораторні випробування системи та її модулів.</td></tr>
<tr><td></td><td></td></tr>
<tr><td>9. Порядок контролю і приймання</td><td colspan="2">Загальні вимоги до приймання розроблюваного виробу:<br>1) Перевірка ходу розробки програмного виробу керівнику робіт виконувати 1 раз в 3 тижні.<br>2) Випробування виробу відповідно до Програми і методики випробувань провести на базі комп'ютерного класу або на базі приватного приміщення.<br>3) Захист розробленого виробу провести на засіданні атестаційної комісії.<br>4) Пояснювальну записку представити на паперових носіях в одному примірнику та в електронному вигляді.</td></tr>
</table>

Виконавець:
студент групи КІ-61

Живага В.В.

_____

Замовник:
канд. тех. наук, старший викладач кафедри електроніки і управляючих систем

Малахова М. О.

_____

Додаток В

Затверджую

_____

«_____»_____2021 р.

**Програма і методика випробувань**
**виробу**
«Система дистанційного моніторингу та керування»

**1 Об'єкт випробувань**

1.1 Найменування випробуваного програмного виробу: система дистанційного моніторингу та керування.

1.2 Область його застосування: побутова, медична та промислова галузі.

**2. Мета випробувань**

Перевірка працездатності виробу та правильності роботи окремих підсистем та системи. Підтвердження характеристик розробленого виробу вимогам, які сформульовані в ТЗ.

**3. Загальні положення**

**3.1 Підстави для проведення випробувань**

Підставою для проведення випробувань є наказ про призначення атестаційної комісії та перевірку даного виробу.

**3.2 Місце і тривалість випробувань**

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу або будь-якого приміщення в період роботи атестаційної комісії.

**3.3 Обсяг випробувань**

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї Програми і методики випробувань.

**3.4 Організації, які беруть участь у випробуваннях**

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

**4. Вимоги до виробу**

4.1. Вимоги до функціональних характеристик: можливість моніторингу навколишнього середовища та керування певними пристроями.

4.2. Вимоги до надійності: забезпечення працездатності системи при несправності другорядного модуля, забезпечення перешкодостійкого обміну даними.

4.3. Вимоги до умов експлуатації: встановлення програмно-апаратних засобів та живлення, під'єднання головного пристрою до локальної мережі, наявність браузера для керування системою.

4.4. Вимоги до складу і параметрів технічних засобів: пристрій з підтримкою підключення до мережі та наявністю браузера, локальна мережа.

4.5. Вимоги до інформаційної та програмної сумісності: підтримка з різними ОС та різними браузерами.

4.6. Вимоги до маркування та упаковки: жорстка упаковка з елементами пінопласту, маркування на українській та англійській мовах.

4.7. Вимоги до транспортування і зберігання: транспортування в упаковці будь-яким способом, температура транспортування/зберігання −10-+20 С°.

4.8. Спеціальні вимоги: відсутні.

**5. Вимоги до документації**

Склад програмної документації, що подається на випробування, включає:

1) Справжнє Технічне завдання на розробку виробу (представлено в Додатку Б до пояснювальної записки до кваліфікаційної роботи).

2) Програму і методику випробувань розробленого програмного вироби Програму і методику випробувань розробленого виробу (представлено в Додатку В до пояснювальної записки до кваліфікаційної роботи).

3) Опис виробу (представлено в Розділі 3 до пояснювальної записки до кваліфікаційної роботи).

4) Фрагменти тексту програми (представити у Додатках до пояснювальної записки до кваліфікаційної роботи).

**6. Засоби і порядок випробувань**

**6.1 Засоби випробувань**

Засоби випробувань представлено в комп'ютерному класі без спеціальних вказівок.

**6.2 Порядок проведення випробувань**

Як правило, випробування проводяться в два етапи:

- ознайомчий (1-й етап);
- власне випробування програмного виробу (2-й етап).

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1) Перевірку комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в ТЗ документації.

2) Виконання тесту:

Тест 1: Встановити систему та перевірити правильність з'єднань (Пройдено).

Тест 2: Змінити мову, натиснувши на потрібну, у верхньому кутку веб-сторінки (Пройдено).

Тест 3: Перевірити можливість перемикання розділів (Indicators, Devices тощо) веб-сторінки (Пройдено).

3) Якість програмної документації перевіряється на відповідність до стандартів ISO/IEC 26514:2008 – Пройдено вдало.

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

1) Перевірку відповідності технічних характеристик програми вимогам технічного завдання.

2) Перевірку ступеня виконання функціональних вимог до програми.

3) Методику проведення перевірок:

Тест 1: Встановити систему віддаленого доступу та налаштувати мережу.(Пройдено)

Тест 2: З'єднати усі потрібні підсистеми та увімкнути живлення. (Пройдено)

Тест 3: Порядок проведення випробувань: (Пройдено)

- Запустити браузер та встановити з'єднання с сервером через локальну мережу.
- Перевірити правильність показників моніторингу на вкладці «Indicators».
- Перевірити можливість увімкнути/вимкнути пристрій за допомогою веб-додатку.

4) Усі тести пройдено вдало, що означає вдале проходження випробування.

Виконавець:
студент групи КІ-61 Живага В.В. _____

## Додаток Г.
## Використання інтерфейсу SPI для ОК

```
#include <cstdlib>
#include <iostream>
#include <stdint.h>

#include <RF24/nRF24L01.h>
#include <RF24/RF24.h>

using namespace std;

#define BCM_PIN 9
#define SPI_DEV 0
#define MAX_LEN 30

uint8_t saveData[MAX_LEN];
uint8_t pinNumber;
uint8_t loadSize;

uint8_t pinAddress[4][3] = {          // номера труб

{ '0', 'P', 'I', 'P', 'E' }};

int main() {

  RF24 radioLisn(BCM_PIN, SPI_DEV);  //Создаём объект
  radioLisn.begin(); //Инициируем работу nRF24L01+
  radioLisn.setAutoAck(1); // режим подтверждения приёма
  radioLisn.setRetries(0, 15);
  radioLisn.enableDynamicPayloads();
  for (int i = 0; i < 6; i++) {
    radioLisn.openReadingPipe(i, pinAddress[i]);
  }
  radioLisn.setChannel(0x70);
```

```
radioLisn.setPALevel(RF24_PA_MAX);// Указываем мощ. передатч.
radioLisn.setDataRate(RF24_1MBPS);// Указ. скор. пер. данных
radioLisn.startListening();//начинаем прослушивать
radioLisn.printDetails();
cout << "Start" << endl;
while (true) {
  /* Если в буфере имеются принятые данные */
  if (radioLisn.available(&pinNumber)) {
    loadSize = radioLisn.getDynamicPayloadSize();
    /*
     * Читаем данные в массив data и указываем сколько байт читать
     */
    radioLisn.read(&saveData, loadSize);
    cout << "Pipe: " << (int) pinNumber << "; ";
    cout << "Size: ";
    printf("%02d", (int) loadSize);
    cout << "; ";
    cout << "Data: [";
    for (uint8_t i = 0; i < loadSize; ++i) {
      if (i == 0) {
        printf("%02X", saveData[i]);
      } else {
        printf(", %02X", saveData[i]);
      }
    }
    cout << "]" << endl;
  }
  __msleep(10);
}
return 0;
}
```

**Додаток Д.**

## Використання інтерфейсу SPI для МК

```
#include <nRF24L01.h>
#include <RF24.h>
#include <printf.h>

RF24 radioSent(9, 10);

byte counter = 0;
uint8_t data[30];
uint8_t loadSize;
uint8_t pinAddress[6][5] = {  { '0', 'P', 'I', 'P', 'E' },};
void setup(void) {
  Serial.begin(115200);
  printf_begin();
    radioSent.begin();
    radioSent.setAutoAck(1);
    radioSent.enableDynamicPayloads();
    radioSent.setChannel(0x70);
    radioSent.setDataRate(RF24_1MBPS);
    radioSent.setPALevel(RF24_PA_MAX);
    radioSent.openWritingPipe(pipesAddress[0]);
    radioSent.stopListening();
    radioSent.printDetails();
}

void loop(void) {
  loadSize = tempDate;//підготовка к отправке
  for (uint8_t i = 0; i < loadSize; i++) {
    data[i] = i;
  }
  Serial.print("Sent:");
  for (uint8_t i = 0; i < loadSize; i++) {
```

```
    Serial.print(" ");
    Serial.print((int) data[i]);
  }
  Serial.print("; ");
  if (radioSent.write(data, loadSize)) {
    Serial.println("Данные приняты приёмником.");
  } else {
    Serial.println("Na date.");
  }

  delay(1000);
}
```

**Додаток Е.**

## Використання інтерфейсу $I^2C$ для ОК

```
import java.util.HashMap;
import java.util.Map;

import com.pi4j.io.i2c.I2CBus;
import com.pi4j.io.i2c.I2CDevice;
import com.pi4j.io.i2c.I2CFactory;
import com.pi4j.platform.Platform;
import com.pi4j.platform.PlatformManager;

public class I2DHT12Read {
  private static final int DHT12_ADDRES = 0xB8;

  private static final int DHT12_HUMID_ADDRES = 0x00;
  private static final int DHT12_HUMID_FRACTION_ADDRES = 0x01;
  private static final int DHT12_TEMPER_ADDRES = 0x02;
  private static final int DHT12_TEMPER_FRACTION_ADDRES = 0x03;
private static final int DHT12_CONTROL_ADDRES = 0x04;

  public static void main(String[] args) throws Exception {

    PlatformManager.setPlatform(Platform.ORANGEPI);
```

```
I2CBus i2cDateBus = I2CFactory.getInstance(I2CBus.BUS_0);
I2CDevice i2cDevice = i2c.getDevice(DHT12_ADDRES);

int humid = device.read(DHT12_HUMID_ADDRES);
int humidFract = device.read(DHT12_HUMID_FRACTION_ADDRES);
int temp = device.read(DHT12_TEMPER_ADDRES);
int tempFract = device.read(DHT12_TEMPER_FRACTION_ADDRES);
int conrtrolSum = device.read(DHT12_CONTROL_ADDRES);
int nint;

double temperature = 0.25 * temp;

System.out.println("t = " + temperature + " °C");

Thread.sleep(1000);
```

**Додаток Ж.**

## Програмна реалізація TCP/IP серверу.

```
#include "tcp_serv.h"
unsigned int numEth = 0;
//--------------для работы по TCP------------------//
int socket_desc , client_sock , c , read_size;
struct sockaddr_in server , client;
char client_message[BUF_SIZE];
int sock;
//----------------структура для клиентов-----------//
struct Client
{
      char clientAddr[20]; //IP клиента
      int fd; //номер подключения клиента
};

void initTctServ(unsigned int num)
{
      int id2, result;
      pthread_t thread2;

      numEth=num; //запоминаем соединение
      //------------------создание потока 2-------------------------//
      //создаю поток для отлова новых подключений
      id2=2; //2-я переменная, для ID потока 2
      result = pthread_create(&thread2, NULL, thrd_2, &id2);
      if (result==0) print_mes("thread TCP create\n",1,0); else
            print_mes("thread TCP create error\n",1,1);
}

/*
 * поток для работы TCP сервера
 */
void * thrd_2(void *arg)
{
```

```
    int socket_desc , client_sock , c;
    struct sockaddr_in server , client;
    //Create socket
    socket_desc = socket(AF_INET , SOCK_STREAM , 0);
    if (socket_desc == -1)
    {
        print_mes("Could not create TCP socket\n",1,0);
    }
    puts("TCP socket created\n");
    //Prepare the sockaddr_in structure
    server.sin_family = AF_INET;
  // server.sin_addr.s_addr = INADDR_ANY;
    server.sin_port = htons(PORT_TCP_SERV);
    if (numEth==1) inet_aton(IP_eth0, &server.sin_addr); else
    if (numEth==2) inet_aton(IP_eth1, &server.sin_addr);
    //Bind
    if( bind(socket_desc,(struct sockaddr *)&server , sizeof(server))
< 0)
    {
        //print the error message
        print_mes("bind TCP failed. Error\n",1,1);
        pthread_exit(NULL); //завеершаю поток
    }
    puts("bind done\n");
    //Listen
    listen(socket_desc , 3);
    //Accept and incoming connection
    puts("Waiting for incoming connections...\n");
    c = sizeof(struct sockaddr_in);
    pthread_t thread_id;
    while( (client_sock = accept(socket_desc, (struct sockaddr
*)&client, (socklen_t*)&c)) )
    {
        puts("Connection accepted\n");
        //создаю новые потоки для каждого клиента
```

```
            if( pthread_create( &thread_id , NULL ,  connection_handler
, (void*) &client_sock) < 0)
            {
                  print_mes("could not create thread\n",1,1);
                  pthread_exit(NULL); //завеершаю поток
            }
            puts("Handler assigned\n");
      }
      if (client_sock < 0)
      {
            perror("accept failed");
            pthread_exit(NULL); //завеершаю поток
      }
      return 0;
}



/*
 * This will handle connection for each client
 * */
void *connection_handler(void *socket_desc)
{
    //Get the socket descriptor
    sock = *(int*)socket_desc;
    int read_size;
   // char *message;
    char client_message[BUF_SIZE];



    //Receive a message from client
    while( (read_size = recv(sock , client_message , BUF_SIZE , 0)) >
0 )
    {
     if (read_size <= 2048) decodeTcpPacket(client_message, sock);
//обработка пакета
            memset(client_message, 0, BUF_SIZE);
```

```
    }

    if(read_size == 0)
    {
     print_mes("Client disconnected\n",1,0);
        fflush(stdout);
    } else
    if(read_size == -1)
    {
     print_mes("recv failed\n",1,1);
    }
    return 0;
}
```