Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
Розроблення організаційного та програмного забезпечення для супроводження автоматизованих систем оповіщення на виробництві
(тема)

Виконав:
студент *II курсу, групи КІТПВм 21-1*
Шостенко С. С.
(прізвище, ініціали)
Спеціальності *151 – Автоматизація та комп'ютерно-інтегровані технології*
(код і повна назва спеціальності)
Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Комп'ютерно-інтегровані технологічні процеси і виробництва*
(повна назва освітньої програми)
Керівник проф. Филипенко О. І
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри КІТАМ _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2022р.

Я, як студентка ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавала і не одержувала недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«02» грудня 2022 р. *Шостенко С.* С.

Харківський національний університет радіоелектроніки

Факультет Автоматизація та комп'ютерно-інтегровані технології
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
Рівень вищої освіти другий (магістерський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми Освітньо-професійна
Освітня програма 151 Комп'ютерно-інтегровані технологічні процеси і виробництва
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАМ _____
(підпис)
«_»_____20_ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Шостенко Світлана Сергіївна
(прізвище, ім'я, по батькові)

1. Тема роботи *«Розроблення організаційного та програмного забезпечення для супроводження автоматизованих систем оповіщення на виробництві»*
Затверджена наказом по університету від «07» листопада 2022 р. №1464 Ст
2. Термін подання студентом роботи до екзаменаційної комісії «07» грудня 2022 р.
3. Вихідні дані до роботи: *Об'єкт дослідження – процес розроблення організаційного та програмного забезпечення для супроводження автоматизованих систем оповіщення на виробництві. Предмет дослідження – організаційне та програмне забезпечення для супроводження автоматизованих систем оповіщення на виробництві. Технічне забезпечення датчики, пристрій ТТА-08, мобільний телефон. Перелік використовуваних програмних засобів Visual Studio Code, Firebase Console.*

4. Перелік питань, що потрібно опрацювати в роботі: *Вступ, аналіз предметної області, огляд автоматизованих систем оповіщення на виробництві, моделювання автоматизованої системи оповіщення на виробництві стійкість автомотазиваної системи, проектування програмного забезпечення, аналіз існуючих фреймворків, платформа*

розробки, аналіз архітектурних патернів, розроблення програмного забезпечення

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри): *демонстраційний матеріал, представлений у форматі презентації PowerPoint (*.ppt) на аркушах формату А4 (16 сторінок)*

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Приміт-ка
1	Отримання, аналіз завдання, уточнення плану роботи роботи	01.09.22	Виконано
2	Аналіз предметної області	08.10.22	Виконано
3	Математична модель системи	15.10.22	Виконано
4	Стійкість автоматизованої системи	20.11.22	Виконано
5	Проектування програмного забезпечення	25.11.22	Виконано
6	Оформлення пояснювальної записки	10.12.22	Виконано
7	Подання закінченої роботи науковому керівникові	15.12.22	Виконано
8	Усунення зауважень наукового керівника	16.12.22	Виконано
9	Подання роботи на рецензування	17.12.22	Виконано
10	Підготовка презентації	18.12.22	Виконано
11	Попередній захист	19.12.22	Виконано
12	Подання роботи до екзаменаційної комісії	20.12.22	Виконано

Дата видачі завдання _____

Студент _____ Шостенко С. С.
(підпис) (прізвище, ініціали)

Керівник роботи _____ проф. Филипенко О. І
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 60 с., 25 рис., 2 дод., 25 джерел.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, FLUTTER, DART, АВТОМАТИЗОВАНА СИСТЕМА, СИСТЕМА ОПОВІЩЕННЯ.

Об’єкт дослідження – процес розроблення функціонального, математичного та програмного забезпечення автоматизованих систем оповіщення на виробництві.

Предмет дослідження – автоматизовані системи оповіщення на виробництві.

Мета роботи – підвищення якості роботи системи оповіщення, для людей з вадами зору та слуху, шляхом розроблення функціонального, математичного та програмного забезпечення автоматизованих систем оповіщення на виробництві.

Методи дослідження – експеримент, спостереження, аналіз, індукція, дедукція, математичний аналіз.

Було проведено дослідження, що описує проблематику використання системи мовного сповіщення у якості системи оповіщення на виробництві. Під час дослідження було вирішено ці питання через організаційного та програмного забезпечення для супроводження автоматизованих систем оповіщення на виробництві. Саме така система вирішить питання недоступності голосового сповіщення за для людей з вадами слуху через світлове повідомлення о екстреній ситуації.

Отримані результати апробовані на 2-х міжнародних науково-практичних конференціях [1-2], пов’язані із сферами автоматизації систем оповіщення.

ABSTRACT

Explanatory note: 60 pages, 25 figures, 2 additional, 25 sources.

SOFTWARE, FLUTTER, DART, AUTOMATED SYSTEM, ALERT SYSTEM.

The object of the research is the process of developing functional, mathematical and software of automated notification systems in production.

The subject of the research is automated notification systems in production.

Research methods - experiment, observation, analysis, induction, deduction, mathematical analysis.

A study was conducted that describes the problems of using a voice notification system as a notification system at work. During the study, these issues were resolved through organizational and software support for automated notification systems in the workplace. It is this system that will solve the issue of the unavailability of voice notification for people with hearing impairments due to a light message about an emergency situation.

The obtained results were tested at 2 international scientific and practical conferences [1-2] related to the areas of automation of notification systems.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АУПС – автоматичні установки пожежної сигналізації;

КІТАМ – комп’ютерно-інтегрованих технологій, автоматизації та мехатроніки;

СКБ – студентське конструкторсько-технологічне бюро;

СОВ – система оповіщення на виробництві;

СМО – система масового обслуговування.

BLoC – business logic component;

UI – user interface.

ВСТУП

У сучасному світі автоматизована система оповіщення являє собою сукупність процесів, технологій, алгоритмів дій, а також організаційно і технічно поєднаних електронних комунікацій, програмних і технічних засобів оброблення та передачі інформації, що призначені для своєчасного доведення сигналів та інформації до органів виконавчої влади, органів місцевого самоврядування, органів управління і сил цивільного захисту, підприємств, про загрозу виникнення або виникнення надзвичайних ситуацій.

Найчастіше у якості автоматизованої системи центрального оповіщення використовують голосову систему. Як правило, така система складається з мікшера-підсилювача, мікрофонів та гучномовців, що підбираються під різні типи приміщень.

Ця система має велику перевагу у вигляді швидкого сповіщення усіх працівників не залежно від їх місці знаходження. Проте проектування та інтегрування таких систем, займає багато часу. Та якщо підприємство залишається без струмового живлення, то система більш не зможе функціонувати. Також важливо враховувати що для працівників з вадами слуху ця система є неможливою.

За для вирішення цих проблем пропонується розробка організаційного та програмного забезпечення для супроводження автоматизованих систем оповіщення на виробництві, який би виконував роль системи сповіщення. Завдяки голосовим та світловим сповіщенням усі працівники будуть сповіщені о екстреній ситуації. Цей застосунок можливо буде використовувати на виробничих підприємствах та й в будівлях з підвищеною загрозою.

Об’єкт дослідження – процес розроблення функціонального, математичного та програмного забезпечення автоматизованих систем оповіщення на виробництві.

Предмет дослідження – автоматизовані системи оповіщення на виробництві.

Мета роботи – підвищення якості роботи системи оповіщення, для людей з вадами зору та слуху, шляхом розроблення функціонального, математичного та програмного забезпечення автоматизованих систем оповіщення на виробництві.

Методи дослідження – експеримент, спостереження, аналіз, індукція, дедукція, математичний аналіз.

Під час проектування мобільного додатку, згідно тематики кваліфікаційної роботи були написані наступні наукові публікації [1-2], пов’язані із сферами автоматизації систем оповіщення.

Кваліфікаційна робота оформлена згідно з вимогами ДСТУ 3008 2015 [3], а також з рекомендаціями з підготовки і оформлення кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти [4].

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести огляд автоматизованих систем оповіщення на виробництві;

– розробити схему та алгоритмічне забезпечення фунавтоматизованої системи оповіщення на виробництві;

– розрахувати математичну модель автоматизованої системи оповіщення на виробництві;

– розрахувати стійкість автоматизованої системи;

– провести аналіз існуючих фреймворків;

– обрати платформу розробки;

– провести аналіз архітектурних патернів;

– розроблення програмного забезпечення.

1 АНАЛІЗ АВТОМАТИЗОАНИХ СИСТЕМ СПОВІЩЕННЯ

1.1 Огляд автоматизованих систем оповіщення на виробництві

Автоматизована система оповіщення являє собою сукупність процесів, технологій, алгоритмів дій, а також організаційно і технічно поєднаних електронних комунікацій, програмних і технічних засобів оброблення та передачі інформації, що призначені для своєчасного доведення сигналів та інформації з питань цивільного захисту до органів виконавчої влади, органів місцевого самоврядування, органів управління і сил цивільного захисту, підприємств, установ, організацій та населення про загрозу виникнення або виникнення надзвичайних ситуацій.

Але якщо розглядати у ракурсі виробничих підприємств (рис. 1.1), то локальна система оповіщення — автоматизована система оповіщення на об’єкті підвищеної небезпеки призначена для оповіщення персоналу об’єкта та населення в разі загрози виникнення та під час виникнення надзвичайних ситуацій, у результаті яких зона можливого ураження також досягає територій інших об’єктів або заселених територій.

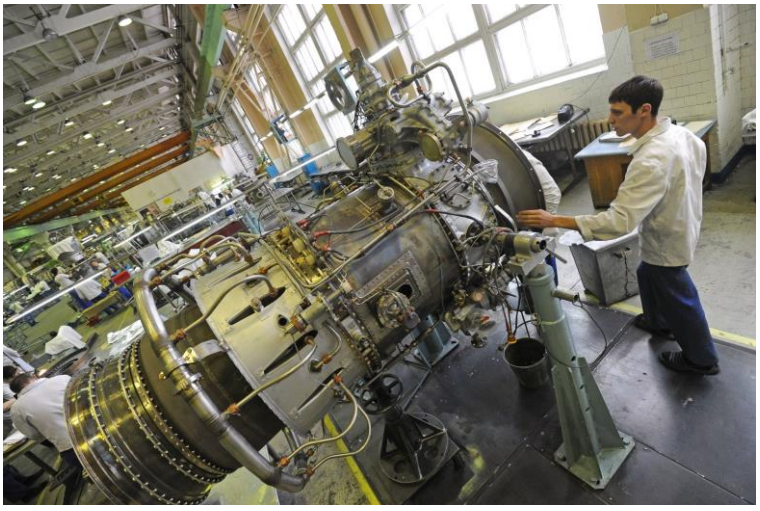
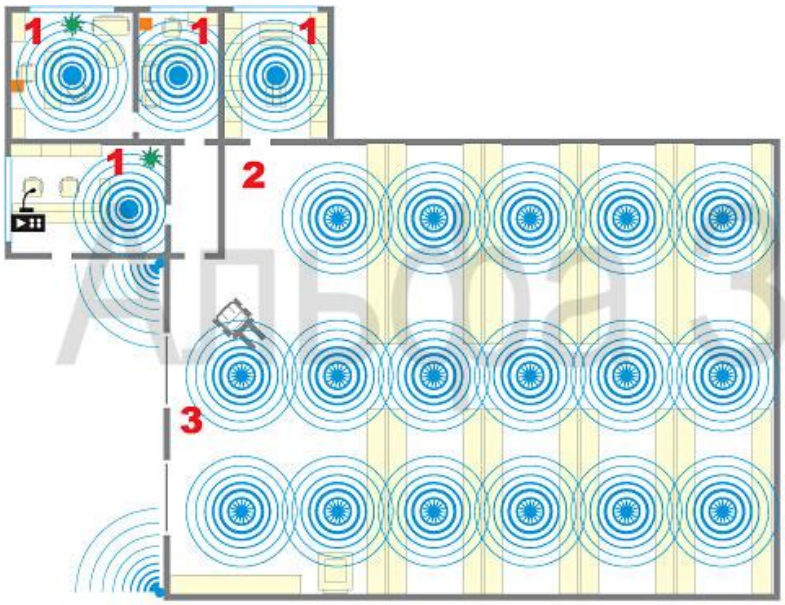


Рисунок 1.1 – Виробниче підприємство

Найчастіше у якості автоматизованої системи центрального оповіщення використовують голосову систему (рис. 1.2). Як правило, така система складається з мікшера-підсилювача, мікрофонів та гучномовців, що підбираються під різні типи приміщень. Стелі адміністративних приміщень часто дозволяють використовувати гучномовці, що вбудовуються. Для таких приміщень ідеально підійдуть компактні гучномовці. Розмір і невелика вага дозволяють вбудовувати їх у найтендітніші плити підвісних стель [5].

В одному з адміністративних приміщень слід розташувати мікшер-підсилювач з підключеними мікрофонами. Саме звідси вестиметься передача фонової музики та голосових повідомлень. Фонову музику можна транслювати зонами одночасно або вибірково, але при мовному повідомленні диспетчера трансляція музики переривається. Це дуже зручно, адже часто одночасно доводиться коригувати роботу кількох зон [6-7].



1 – адміністративна частина, 2 – складське приміщення, 3 – зона навантаження\вивантаження

Рисунок 1.2 – Схема голосового сповіщення

Кожне адміністративне приміщення має бути оснащене регулятором гучності, що дозволяє індивідуально регулювати звук або вибірково вимикати його.

Система пожежного оповіщення може бути інтегрована із системою голосового оповіщення. При настанні надзвичайної ситуації вона може працювати в автоматичному та напівавтоматичному режимі [8]. Автоматичний режим передбачає пріоритетну трансляцію сигналу тривоги зонами у вигляді записаного повідомлення, напівавтоматичний режим – передачу голосового повідомлення диспетчера з мікрофонної консолі.

Проектування озвучування самого складського приміщення потребує врахування багатьох чинників. Це і площа, і об'єм приміщення, висота та матеріал стель. Саме ці параметри визначають, яка має бути потужність, кількість та тип гучномовців. Як правило, особливості стель не дозволяють використовувати в складському приміщенні вбудовані гучномовці. Тут краще використовувати підвісні динаміки. Високі стелі сучасних складських комплексів дозволяють використовувати менше підвісних гучномовців, що мають високу потужність [9-10].

Також важливо враховувати при проектуванні тип складського обладнання та його розташування. Система голосового сповіщення складського приміщення з металевими стелажами відрізнятиметься від складу з палетними та мізонінними стелажами. Це має враховуватися під час створення проекту.

Для зони навантаження-розвантаження застосовують гучномовці рупорного типу. Вони розроблені спеціально для великих просторів, сирих та холодних приміщень. Їхні конструкторські особливості забезпечують можливість трансляції гучного, розбірливого, без перешкод, звуку. Рупорні гучномовці кріпляться на стовпи освітлення, стіни будівлі та мають високий рівень захисту від впливу навколишнього середовища.

Система має велику перевагу у вигляді швидкого сповіщення усіх працівників не залежно від їх місці знаходження. Проте проектування та

інтегрування таких систем, займає багато часу. Та якщо підприємство залишається без струмового живлення, то система більш не зможе функціонувати. Також важливо враховувати що для працівників з вадами слуху ця система є неможливою.

1.2 Технічне завдання до автоматизованої системи оповіщення на виробництві

Після проведеного аналізу аналогічних систем, наведення їх недоліків та переваг, стає зрозуміло, що при побудові автоматизованої системи оповіщення необхідно:

– розробити схему атоматизованої системи оповіщення на виробництві;

– розрахувати математичну модель автоматизованої системи;

– визначити стійкість автоматизованої системи;

– побудувати алгоритмічне забезпечення автоматизованої системи оповіщення на виробництві;

– проаналізувати існуючі фреймворкі, платформи розробки та архітектурні патерни;

– розробити програмне забезпечення та протестувати.

1.3 Висновки до першого розділу

У результаті першого розділу було вирішено, що за для вирішення цих проблем пропонується розробка мобільного додатку, який би виконував роль системи сповіщення. Завдяки голосовим та світловим сповіщенням усі працівники будуть сповіщені о екстреній ситуації та завдяки схемі евакуації, яка буде виводитись на екран людина зрозуміє куди їй необхідно рухатись.

2 МОДЕЛЮВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ОПОВІЩЕННЯ НА ВИРОБНИЦТВІ

2.1 Розробка схеми атоматизованої системи оповіщення на виробництві

Схема на рис. 2.1 зображує автоматизовану систему оповіщення.

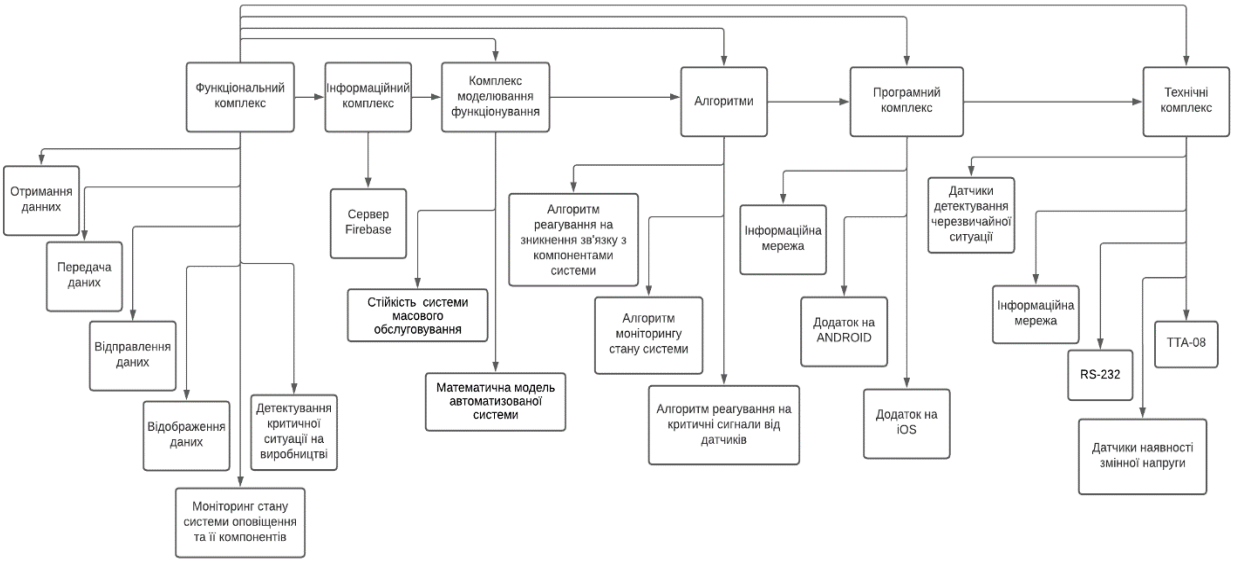


Рисунок 2.1 – Схема автоматизованої системи сповіщення

Функціональний комплекс визначає та формулює всі функції, які потрібно реалізовувати автоматизованій системі.

Інформаційний комплекс у данному випадку це Firebase, він виконує роль серверної частини, яка виконує роль зв’язуючого елементу між додатком та технічним комплексом.

Інформаційний комплекс визначає джерела інформації вхідної та вихідної інформації.

Комплекс моделювання функціонування дає математичний опис принципів функціонування системи та моделювання процесу функціонування по цим моделям.

Алгоритми зображують принцип роботи при системи за заданих умов.

Програмний комплекс у даному випадку це мобільний додаток, він виконує роль сповіщувача, для працівників на виробництві. Саме завдяки ньому можливо дізнатись про критичну ситуацію, про стан автоматизованої системи. Важливою перевагою цього додатку є його кросплатформність.

Технічний комплекс, зображує необхідне фізичне обладнання для системи.

Датчики детектування черезвичайної ситуації у данному випадку це пристрої, які здатні розпізнати критичну ситуацію у приміщені, такі як датчики диму, температури, тиску, тощо.

RS-232 – це інтерфейс передачі між двома пристроями на відстань до 15 м. Використовується для підключення до обчислювальних машин різного обладнання, невибагливого до швидкості обміну.

Датчики наявності змінної напруги необхідні для детектування живлення по всій системі.

Модуль управління ТТА-08 (рис. 2.2) – це пристрій призначений для дистанційного моніторингу (через мережу Ethernet) віддаленого обладнання та об'єктів [6-7].

Можливе підключення датчиків та сенсорів для моніторингу кліматичних параметрів (температура, вологість), а також стандартного охоронного обладнання (датчики руху, датчики розбивання скла, датчики відкриття вікон/дверей, датчики диму, датчики затоплення).

Пристрої, що можуть бути підключені:

– чотири шлейфи для підключення приладів, а також вихід живлення приладів охоронної/пожежної сигналізації (10..12 В, 100 мА або 10..15 В 750 мА);

– вхід виносного датчика температури;

– вхід виміру напруги 0..3 В (датчик вологості);

– вхід виміру напруги 0..90 В (акумуляторна батарея);

– два входи підключення датчиків наявності ~220 В;

– вхід підключення акумуляторної батареї з напругою 12 В.

Рисунок 2.2 – Пристрій ТТА-08

У базовій комплектації пристрій дистанційного моніторингу та контролю датчиків дозволяє підключити охоронне обладнання (датчик руху, розбиття скла, відкриття вікон/дверей, датчик диму та затоплення), а також підключити датчик вимірювання температури та датчик вологості.

Застосування датчиків вимірювання температури та вологості дозволяє побудувати на базі пристрою дистанційного моніторингу систему контролю кліматичних параметрів у різних приміщеннях.

А застосування спеціалізованого датчика вимірювання температури дозволяє побудувати систему багатоточкового контролю/моніторингу температури з кількістю точок контролю від кількох одиниць до кількох сотень.

Також пристрій дистанційного моніторингу та контролю здатен не тільки перевіряти стан підклчених датчиків а й перевіряти наявність напруги та заряду батареї (рис. 2.3). Пристрій має два входи для підключення датчиків наявності змінної напруги 220 В та один вхід для вимірювання напруги акумуляторної батареї (табл. 2.1).

Вимірювання напруги акумулятора здійснюється в діапазоні від 0 до 90 В постійного струму з точністю вимірювання +/- 1 В.

Передбачена можливість через програмне забезпечення пристрою дистанційного моніторингу задати нижній і верхній поріг напруги акумулятора для аварійного інформування через запит на сервер [8-10].

Таблиця 2.1 – Напруга на батареї та контроль входів 220 В

Напруга батареї	
Мінімальне значення для тревоги	9 В
Дійсне значення	14,4 В
Максимальне значення для тревоги	16 В
Контроль входів 220 В	
Вхід 220 В #0	OFF
Вхід 220 В #1	ON

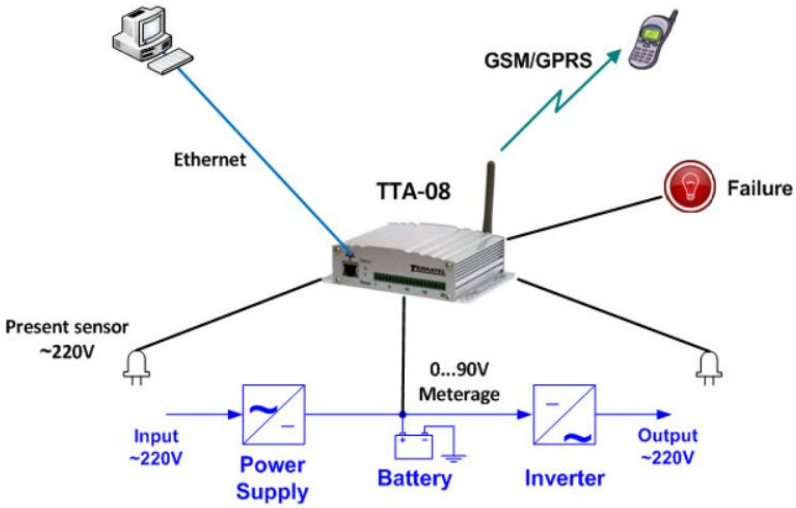


Рисунок 2.3 – Моніторинг живлення системи

Система покликана забезпечити моніторинг акумулятора та наявності мережі 220 В. Система в реальному часі контролює наявність мережі 220 В (2 входи) та вимірює поточне значення напруги акумулятора. У разі відсутності напруги мережі 220 В, або коли поточна напруга акумулятора виходить за межі встановлених нижнього або верхнього порогу напруги акумуляторної батареї, пристрій посилає повідомлення на сервер.

Функціональною особливістю TTA-08 [11] є можливість підключення до мережі Ethernet пристроїв із послідовним інтерфейсом RS-232 та RS-485. (рис. 2.4).

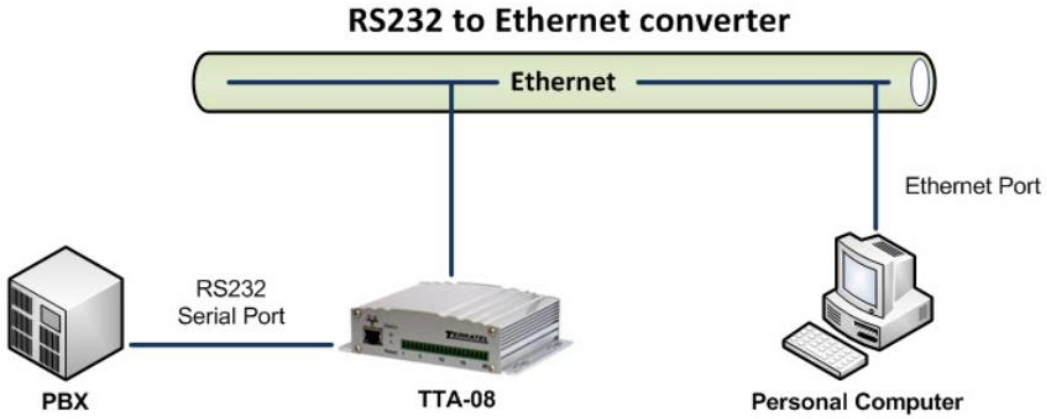


Рисунок 2.4 – Підключення до мережі Ethernet пристроїв із послідовним інтерфейсом RS-232 та RS-485

Наявність зовнішніх гальванічно розв'язаних послідовних інтерфейсів RS-232 та RS-485 дозволяє підключити до локальної мережі (LAN) широкий спектр додаткових пристроїв сторонніх виробників: лічильники електроенергії, зчитувачі смарт-карток та RFID. Пристрій підтримує один послідовний порт.

2.2 Математична модель автоматизованої системи оповіщення на виробництві

Для опису математичної моделі використовується одноканальна система масового обслуговування (СМО) з очікуванням, кінцевим джерелом пріоритетних та непріоритетних заявок, експоненційним вхідним потоком та законом розподілу Ерланга 4-го порядку для тривалості обслуговування [12-15].

У даній моделі канал зв'язку застосовується для обслуговування класів вимог k ($k > 1$), що відрізняються ступенем важливості. Для відмінності

класів за ступенем важливості кожному класу вимог приписується пріоритетний індекс i ($1 \le i \le k$), причому $i = 1$ позначає найвищий ступінь важливості, $i = k$ – найнижчий.

Запропонована математична модель в загальному вигляді описується наступним чином: на один обслуговуючий канал зв'язку з незалежних кінцевих джерел надходять заявки k ($k \ge 2$) типів. Заявки, що надходять із і-го джерела ($1 \le i \le k$) об'ємом N_i, називаються заявками і-го класу. Передбачається, що кожна заявка надходить на обслуговування після перебування у джерелі протягом випадкового часу (із середнім значенням), що визначається експоненційним законом розподілу.

Тривалість обслуговування заявок і-го класу ($1 \le i \le k$) є незалежною, однаково розподіленою випадковими величинами із щільністю $S(t)$. Інформацію про джерела можна записати у вигляді $N_1, N_2, \ldots, Ni, \ldots, Nk$.

Єдиний підхід до аналізу позасистемних пріоритетних дисциплін ґрунтується на понятті циклу обслуговування заявок, тобто періоду часу від моменту надходження на обслуговування і до моменту, коли канал зв'язку звільняється для обслуговування заявок цього класу.

Метод аналізу, що використовується в даному випадку, полягає в дослідженні процесу на періоді зайнятості та в отриманні за допомогою теорії відновлення результатів для вихідного процесу. Тоді дослідження процесу із пріоритетами зводиться до вивчення процесу на циклі обслуговування, який має на один клас заявок менше, ніж вихідний процес. Для моделей $N_1, N_2, \ldots, Ni, \ldots, Nk$ у випадку, коли $k = 2$, тобто з двома класами пріоритетів, заявки першого класу умовно визначено пріоритетними, а другого класу – непріоритетними. Такий підхід не порушує спільності в характері аналізованої математичної моделі та практично встановлює два класи пріоритету для КСМПБ на виробництві.

Для випадкових величин, що характеризують пріоритетні вимоги на циклі обслуговування, вводиться індекс 1, а для непріоритетних вимог – 2.

Імовірність того, що мережа зв'язку вільна та заявки будь-якого пріоритету на обслуговування не надходять, визначається таким виразом:

$$P_0^{(2)} = P_1\left\{1 + T_{П}\left[N_2\lambda_2\sum_{e=0}^{N_2-1}\binom{N_2-1}{e}\frac{1}{\Phi(e)} + N_1\lambda_1\sum_{e=1}^{N_2}\binom{N_2}{e}\frac{(1-в_1(e\lambda_2))}{\Phi(e-1)}\right]\right\}^{-1}, \qquad (2.1)$$

де $\Phi(e) = \left\{\prod_{m=1}^{e}\frac{\overline{f_2}(\lambda_1 + m\lambda_2)}{\left[1-(\lambda_1/(\lambda_1+m\lambda_2))\times\overline{f_1}(m\lambda_2)(1-\overline{f_2}(\lambda_1+m\lambda_2))\right]} \text{ якщо } e \neq 0; \right.$
$\left. 1 \quad \text{якщо } e = 0; \right\}$;

P_0 – гранична ймовірність стану СМО;

m – поточна послідовність чисел від 1 до e;

$\lambda_{1,2}$ – інтенсивність надходження заявок (інтенсивність вхідного потоку викликів);

e – послідовність цілого ряду, що приймає значення 0, 1, 2 і т. д.;

$в_1$ – середня величина часу обслуговування пріоритетних вимог на періоді зайнятості;

P_1 – імовірність того, що мережа зв'язку в момент часу t вільна за умови, що вона обслуговує пріоритетні заявки:

$$P_1 = \left[1 + N_1\lambda_1 T_{П1}\sum_{e=0}^{N_i-1}\binom{N_i-1}{k}\left(\frac{1}{\Phi(e)}\right)\right]^{-1}, \qquad (2.2)$$

де T_{n1} – час обслуговування пріоритетної заявки.

За допомогою формул (2.1) та (2.2) було визначено середню кількість пріоритетних і непріоритетних вимог, що знаходяться в мережі в даний час відповідно:

$$E(M_1) = N_1 - \left[(1/\lambda_1 T_{П1})(1-P_1)\right], \qquad (2.3)$$

$$E(M_2) = N_2 - \left[(1/\lambda_2 T_{П2})(1-P^{(2)}{}_0)\right], \qquad (2.4)$$

де $E(M_1)$ та $E(M_2)$ – середня кількість повідомлень, надійшли за час T_{n1}.

Інтервал розвантаження пріоритетних заявок визначається виразом

$$t_{роз1} = \left[\frac{N_1 \lambda_1 T_{П1}}{N_1 \lambda_1 T_{П1} + \delta} \right] \left[(N_1 - 1) T_{П1} - \frac{1 - \delta}{\lambda_1} + \frac{E(C_2)^2}{2 T_{П1}} \right], \quad (2.5)$$

де $\delta = \left[\sum_{e=0}^{N_1 - 1} \binom{N_1 - 1}{e} \left(\frac{1}{\Phi(e)} \right) \right]^{-1}$

$E(C_2)$ – середня довжина циклу обслуговування непріоритетної вимоги;

C_2 – щільність ймовірності циклу обслуговування непріоритетної вимоги.

Інтервал розвантаження мережі від непріоритетних заявок відповідного кінцевого джерела N_2 визначається такою формулою:

$$t_{роз2} = (1 - P_0^{(2)}) \left[(N_2 - 1) E(C_2) + \frac{E(C_2)^2}{2E(C_2)} - \frac{1}{\lambda_2} \right] + N_2 E(C_2) P_0^{(2)} \left[1 + \frac{\lambda_1 (1 - \overline{в_1}(\lambda_2))}{\lambda_2} \right], \quad (2.6)$$

де $P_0^{(2)}$ – визначається виразом (2.1).

На основі класифікації заявок, що надходять до СОВ за пріоритетами, було побудовано математичну модель мережі, яка дозволяє проводити дослідження функціонування мережі зв’язку автоматизованої системи з метою визначення середньої кількості пріоритетних та непріоритетних вимог залежно від навантаження.

Запропонована математична модель дає можливість дослідити процес обміну інформацією між сервером СОВ та АУПС на виробничих об'єктах. Час обміну інформацією у мережі зв'язку безпосередньо залежить від моменту спрацьовування датчиків на об'єкті.

Для оцінки роботи передачі повідомлень у мережі зв’язку автоматизованої системи вводиться поняття ефективності функціонування мережі.

Для передачі повідомлення інформаційною мережою витрачається певний час передачі повідомлень $T_п$ та деякий непродуктивний час $T_н$ через недостатньо ефективну роботу мережі та час відгуку датчиків та серверного обладнання. Внаслідок постійно змінного характеру роботи каналу зв'язку непродуктивні витрати часу є випадковою величиною [16]. Отже, мережа передачі інформації функціонує більш ефективно у тому випадку, коли при одному і тому ж часі передачі інформації відношення $T_п/(T_п+T_н)$ буде більше. Таким чином, ефективність функціонування інформаційної мережі є математичне очікування випадкової величини відношення часу передачі повідомлень до загального часу доставки інформації у різні моменти часу. Ефективність функціонування є показником якості використання мережі для виконання заданих функцій у мережі та визначається за формулою:

$$E = \sum_{i=0}^{n} P_i \left(\frac{T_{Пi}}{T_{Пi} + T_{Нi}} \right), \tag{2.7}$$

де n – число можливих станів інформаційної мережі;

P_i – граничні ймовірності стану мережі;

$T_{пі}$ – ефективний час передачі повідомлення при і-й заявці;

$T_{ні}$ – непродуктивні витрати часу передачі повідомлення при і-й заявці.

Формула (2.3) може бути використана в якості основної характеристики ефективності функціонування мережі лише у випадках, коли час передачі не нормується.

Ймовірність того, що необхідне повідомлення буде передано протягом заданого часу, може бути названа оперативністю зв'язку Q і визначена як

$$Q = P(T_{Пi} + T_{Нi} \le T_Q), \tag{2.8}$$

де T_Q – задана величина часу передачі зі спілкувань системі зв'язку (критерій оперативності).

У мережі передачі інформації у якості непродуктивних витрат слід приймати час, витрачене на спрацювання датчиків на об'єкті, час передачі інформації від датчика до сервера.

У цьому випадку ефективність функціонування визначатиметься як

$$E = \sum_{i=0}^{n} P_i \left(\frac{T_{Пi}}{T_{Пi} + T_{ОЧi}} \right), \tag{2.9}$$

де $T_{ОЧi}$ – час очікування при і-й заявці (непродуктивних витрат часу);

P_i – гранична вірогідність і-го стану мережі зв'язку.

Якщо прийняти якість та надійність передачі інформації ідеальними [17-19], то відповідно до моделі процесу передачі інформації ефективність функціонування може бути обчислена за допомогою виразу

$$E = P_0 + P_1 + \left(\frac{\overline{T_П}}{\overline{T_П} + \overline{t_{ОЧ}}} \right) \sum_{i=2}^{n} P_i, \tag{2.10}$$

де P_0 – гранична ймовірність стану мережі, коли канал обслуговування вільний;

P_1 – гранична верогідність стану системи, коли канал обслуговування зайнятий, але черги немає (процес обслуговування однієї заявки);

$\overline{T_П}$ – середнє значення (математичне очікування) випадкової величини часу передачі повідомлення;

$\overline{t_{ОЧ}}$ – середнє значення (математичне очікування) випадкової величини часу очікування.

Використовуючи нормувальну умову

$$P_0 + P_1 + \sum_{i=2}^{n} P_i = 1, \tag{2.11}$$

отримуємо

$$E = P_0 + P_1 + \left[\frac{\overline{T_П}(1-(P_0+P_1))}{\overline{T_П} + \overline{t_{ОЧ}}} \right]. \qquad (2.12)$$

У цьому випадку оперативність мережі визначається виразом

$$Q = P_0 + P_1. \qquad (2.13)$$

Підставивши у формули (2.11) та (2.13) значення P_0, P_1 і $\overline{t_{ОЧ}}$, отримаємо формули для обчислення показників ефективності функціонування та оперативності мережі передачі інформації.

Для мережі передачі інформації з розподілом часу обслуговування відповідно до закона Ерланга 3-го порядку величини P_0, P_1, $\overline{t_{ОЧ}}$, згідно з формулами (2.15) і (2.16), визначаються наступним чином:

$$P_0 = \left[1 + Np + Np \sum_{e=1}^{N-1} \binom{N-1}{e} \prod_{m=1}^{e} \frac{\left[(\lambda m + 3\mu)^3 - (3\mu)^3 \right]}{(3\mu)^3} \right]^{-1}, \qquad (2.14)$$

$$P_1 = \frac{N}{(N-1)} P_0 \frac{\left[(\lambda(N-1) + (3\mu)^3 - (3\mu)^3 \right]}{(3\mu)^3}, \qquad (2.15)$$

$$t_{ОЧ} = \frac{N}{\mu \left\{ 1 - \left[1 + Np + Np \sum_{e=1}^{N-1} \binom{N-1}{e} \times \prod_{m-1}^{e} \frac{(\lambda m + k\mu)^k - (k\mu)^k}{(k\mu)^k} \right]^{-1} \right\}} - \frac{1}{\lambda} - \frac{1}{\mu}. \qquad (2.16)$$

Для мережі передачі інформації з експоненціальним законом розподілу часу обслуговування (відповідно до закону Ерланга 1-го порядку) значення P_0, P_1, $\overline{t_{ОЧ}}$, і визначаються відповідно до наступних виразів:

$$\Phi(e) = \begin{Bmatrix} 1/\, p^e e! & якщо \quad e \neq 0; \\ 1 & якщо \quad e = 0; \end{Bmatrix}, \qquad (2.17)$$

$$P_0 = \frac{1}{(1 + Np + Np\sum_{e=1}^{N-1}\binom{N-1}{e}p^e e!)}, \tag{2.18}$$

$$t_{Оч} = \frac{1 + (Np-1)\sum_{e=0}^{N-1}\binom{N-1}{e}p^e e!}{\lambda\sum_{e=0}^{N-1}\binom{N-1}{e}p^e e!} - \frac{1}{\mu}. \tag{2.19}$$

Ефективність функціонування мережі передачі інформації, оперативність зв'язку та середній час очікування залежить від трьох основних параметрів системи зв'язку:

– інтенсивності вхідного потоку λ;

– середнього часу передачі повідомлень $\overline{T_П}$;

– число джерел заявок (об'єктів) N.

2.3 Стійкість автоматизованої системи

В умовах сучасного економічного підйому збільшується актуальність стійкого функціонування автоматизованих систем управління технологічного процесу на виробництвах.

Автоматизовані пожежовибухонебезпечні системи на даний момент грають ключову роль у структурі автоматизованих систем управління. При цьому вони являють собою понад швидкісні підсистеми, які здатні забезпечити розпізнавання надлишкового тиску за 0,1 мкс. Та у цьому випадку здійснити в реальних масштабах часу сповіщення усіх працівників. Через це актуальним стає питання стійкості роботи шини автоматизованої пожежовибухонебезпечної системи, а саме каналу передачі інформації.

За для розрахунку автоматизованої системи управління технологічним процесом використовуються моделі, які базуються на умові, що технічні

характеристики елементів автоматизованої системи не змінні у часі, незалежно від впливаючих факторів. Наприклад, такий параметр як швидкість передачі даних, котрий є головним лімітуючим показником системи. У цьому випадку розрахунок швидкості передачі даних будується на основі множення $P\vartheta$, де P – розрядність системної шини; ϑ – тактова частота шини.

У реальних умовах на продуктивність системної магістралі передачі даних впливають безліч факторів: проводимість матеріалів, яка змінюється з часом, збої, недоліки конструкції та зборки, а також велика кількість інших факторів. По даним з джерела [14] відмінність між теоретичною швидкістю передачі даних та практичній може зменшуватись до 25% від розрахованої величини.

У сучасних автоматизованих систем управління технологічним процесом, створених на єдиній програмно-апаратній базі, при виникненні надзвичайної ситуації можливо виникнення конфлікту між інформаційними потоками різних підсистем. Може виникнути ситуація, коли початкова інформація від високошвидкісній високопродуктивний джерел може бути заблокована менш приорітетною лінією, що приведе до повної відмови системи. Розробник автоматизованої пожежовибухонебезпечної системи за для уникнення цього повинен планувати резервні канали передачі інформації, що тягне за собою збільшення вартості технічного проекту.

За для створення моделі оцінки стійкості функціонування інформаційних каналів автоматизованої пожежовибухонебезпечної системи, яка будується на зміні величини інформаційної ентропії системи, першочергово необхідно отримати характеристичне рівняння системи

$$D(S) = a_0 S^n + a_0 S^{n-1} + \ldots + a_{n-1} S + a_n, \qquad (2.20)$$

де a_0, a_1, …, a_n – коефіцієнти характеристичного рівняння;

S – ентропія системи;

n – кількість підсистем автоматизованої пожежовибухонебезпечної системи.

Відомо, що необхідною умовою стійкості системи будь-якого порядку є позитивність усіх коефіцієнтів характеристичного рівняння (2.20):

$$a_0 > 0,\ a_1 > 0, \ldots\ a_n > 0. \tag{2.21}$$

Дійсно, рівняння (2.21) можна представити у вигляді добутка множників, які містять корні S_1, S_2,…, S_n:

$$a_0(S - S_1)\cdot(S - S_2)\cdot\ldots\cdot(S - S_n) = 0, \tag{2.22}$$

Якщо всі корені характеристичного рівняння будуть від'ємні, то всі множники виразу (2.22) будуть мати вигляд:

$$a_0(S + |a_1|) + (S + |a_2|) + \ldots + (S + |a_n|) = 0, \tag{2.23}$$

де $|a_n|$ – значення коренів.

Після множення (2.23), отримаємо (2.21), у якому усі коефіцієнти будуть визначатися додатними членами $|a_n|$ вираз 2.23, тобто будуть додатними. Для систем в яких одна чи дві підсистеми, необхідна умова стійкості є достатньої умовою.

А для систем в яких три та більше підсистем, умова додатності коефіцієнтів характеристичного рівняння є необхідним проте недостатнім. Для того щоб оцінити стійкість системи, необхідно використовувати алгоритм який базується на дослідження англійського математика Е. Рауса. Для оцінки стійкості побудуємо таблицю. У першому рядку таблиці записано у порядку збільшення індексів коефіцієнтів характеристичного рівняння

(2.21), ті які мають парний індекс: a_0, a_2, a_4, a_6, …, у другому рядку – з непарним індексом: a_1, a_3, a_5, … .

У даному випадку число рядків у таблиці (табл. 2.2) дорівнює ступені характеристичного рівняння (2.21) плюс одиниця.

Таблиця 2.2. – Матриця коефіцієнтів характеристичного рівняння

Коефіцієнт r_i	Рядок	Стовбець			
		1	2	3	4
–	1	$a_0 = c_{11}$	$a_2 = c_{21}$	$a_4 = c_{31}$	…
–	2	$a_1 = c_{12}$	$a_3 = c_{22}$	$a_5 = c_{32}$	…
$r_3 = a_0 / a_1$	3	$c_{13} = a_2 - r_3 a_3$	$c_{23} = a_4 - r_4 c_{33}$	$c_{33} = a_6 - r_3 c_7$	…
$r_4 = a_1 / c_{13}$	4	$c_{14} = a_3 - r_4 c_{23}$	$c_{24} = a_5 - r_4 c_{33}$	$c_{34} = a_7 - r_4 c_{43}$	…
$r_5 = c_{13} / a_{14}$	5	$c_{15} = c_{23} - r_5 c_{23}$	$c_{25} = c_{33} - r_5 c_{33}$	$c_{35} = c_{43} - r_5 c_{43}$	…
…	…	…	…	…	…
$r_i = \frac{c_1, i-2}{c_1, i-1}$	i	$c_{1,\,i} = c_{2,\,i-2} - r_i c_{2,\,i-1}$	$c_{2,\,i} = c_{3,\,i-2} - r_i c_{3,\,i-1}$	$c_{3,\,i} = c_{4,\,i-2} - r_i c_{4,\,i-1}$	…

Тоді умову стійкості можна сформулювати наступним чином: для того щоб система була стійкою, необхідно і достатньо щоб коефіцієнти першого стовбця мали один знак. Даний процес може бути легко автоматизовано в силу простоти його формулювання.

Оцінювати роботи системи необхідно при моделюванні інформаційного впливу. Припустимо, що в автоматизованій пожежовибухонебезпечній системі одночасно діють декілька інформаційних каналів. У цьому випадку для автоматизованої пожежовибухонебезпечної системи, яка побудована по дискретному принципу, величина ентропії $S_д$ визначається за формулою:

$$S_{\partial} = -\sum_{i=1}^{n} p_i \log_2 p_i, \qquad (2.24)$$

де p_i – вірогідність появи сигнала в *i*-м каналі.

Відомо, що в автоматизованій пожежовибухонебезпечній системі завжди задіяно хоча б один інформаційний канал. У цьому випадку вірогідність появи сигнала у даному інформаційному каналі дорівнює одиниці. Якщо сигнали з інших каналах відсутні, то вірогідність появи сигнала в них дорівнює нулю, тобто за для заздалегідь відомого інформаційного повідомлення вираз (2.24) буде складатись з складових двох типів: $1 \cdot \log_2 1$, або $0 \cdot \log_2 0$. Складова першого типу дорівнює нулю, тому $\lim_{x \to 0}(x \cdot \log_2 x) = 0$. Виходячи з цього виразу ентропія заздалегідь відомого повідомлення дорівнює нулю.

Далі необхідно визначити яке максимальне значення може приймати доданок $-p_i \log_2 p_i$. Для цього було продиференційовано рівняння (2.24) та цю похідну прирівняно до нулю.

$$(d / dp_i)(-p_i \log_2 p_i) = -p_i \log_2 p_i - p_i - p_i (1/ p_i) \log_2 e = 0, \qquad (2.25)$$

тобто $p_i = 1/e.$

Так як складова виразу (2.24) не перевищує значення $(1/e)\log_2 e,$ то з цього виходить що ентропія автоматизованою пожежовибухонебезпечної системи є величина кінцева при будь-якій кількості інформаційних підсистем. Зрозуміло, що при подібному розподілу вірогідності p_i ентропія приймає максимальне значення, що корелює з результатом Е. Рауса. Далі використовуючи метод невизначених множників Лагранжа найдемо максимум функцій у вигляді виразу (2.27), складеної з урахуванням додаткової умови

$$\sum_{i=1}^{n} p_i = 1, \tag{2.26}$$

$$F = -\sum_{i=1}^{n} p_i \log_2 p_i - \lambda \sum_{i=1}^{n} p_i. \tag{2.27}$$

Після диференціювання (2.27) по p_i далі прирівняно до нуля, отримаємо рівняння (2.28) та (2.29):

$$\frac{dF}{dp_i} = -\log_2 p_i - (\frac{1}{p_i}) p_i \log_2 e - \lambda = 0, \tag{2.28}$$

$$-\log_2 p_i = \log_2 e + \lambda. \tag{2.29}$$

Аналізуючи вираз (2.27), можна зробити висновок, що вірогідність p_i не залежить від змінної підсумування, що може бути тільки у тому випадку, якщо усі вірогідності тотожні між собою: $p_1 = p_2 = \ldots = p_n = p = 1/n$

$$S_o^{\max} = -\sum_{i=1}^{n} \frac{1}{n} \log_2 \frac{1}{n} = \log_2 n. \tag{2.30}$$

Відомо, що ентропія прагне до свого максимального значення. Базуючись на цьому, зрозуміло що ентропія системи досягає максимального значення, коли вірогідність появи інформаційних сигналів тотожні.

Необхідно виділити, що формула (2.24) коректна в ситуації, коли заздалегідь невідомо, яке повідомлення буде передано у той чи інший момент часу, тобто повідомлення на сервер випадкове. Прикладом використання такої ситуації може бути автоматизованою пожежовибухонебезпечної системи, у котрій задіян єдиний інформаційний канал. При цьому інформаційний сигнали надходять до адресату незалежно одне від одного.

У випадку створення автоматизованої пожежовибухонебезпечної системи, у котрій ряд сигналів залежний один від одного, величина інформаційної ентропії визначається в залежності відмінної від (2.24).

Тоді система сигналів (X, Y) розглянемо як один сигнал, а безліч станів такого сукупного сигналу зобразимо двома парами виду (x_i, y_j) де i=1, …, j=1, …, n. При відомих вірогідностях $p(x_i, y_j)$ поява пари (x_i, y_j) ентропія системи сигналів (X, Y) визначається виразом

$$S(X,Y) = -\sum_{i=1}^{n} p(x_i, y_j) \log_2 p(x_i|y_j). \quad (2.31)$$

Згідно теоремі множення вірогідностей маємо:

$$p(x_i, y_j) = p(y_i) p(x_i|y_j), \quad (2.32)$$

$$p(x_i, y_j) = p(x_i) p(y_j|x_i), \quad (2.33)$$

де $p(x_i), p(y_i)$ – вірогідності появи сигналів згідно $x_i \in X, y_i \in Y$;

$p(x_i|y_j)$ – умовна вірогідність появи елемента $x_i \in X$ при умові, що вже існує сигнал $y_j \in Y$;

$p(y_j|x_i)$ – умовна вірогідність появи сигнала $y_j \in Y$ при умові, що вже існує сигнал $x_i \in X$.

Підставляючи у (2.9) вираз (2.10), отримаємо формулу

$$S(X,Y) = -\sum_{i=1}^{n} \sum_{j=1}^{n} p(y_i) p(x_i|y_j) \cdot \log_2 [p(y_j) p(x_i|y_j)]. \quad (2.34)$$

Враховуючи, що $\log_2 (p(y_j) p(x_i|y_j)) = \log_2 p(y_j) + \log_2 p(x_i|y_j)$, останній вираз запишемо у вигляді:

$$S(X,Y) = -\sum_{j=1}^{n} p(y_i)\log_2 p(y_j)\sum_{i=1}^{n} p(x_i|y_j) - \sum_{j=1}^{n} p(y_i)\sum_{i=1}^{n} p(x_i|y_j)\log_2 p(x_i|y_j). \quad (2.35)$$

Відповідно до (2.24) перша сума (2.32) є ентропією сигнала Y, а друга сума другого члена (2.32) можна розглядати як ентропію сигнала X при умові, що другий сигнал Y отримав конкретне значення Y_i. Цю умовну ентропію відносно елемента Y_i можна назвати приватною умовою ентропії, яку позначимо як $S(X|y_j)$.

Аналогічний вираз можна отримати за для $S(Y|x_i)$.

Необхідно зобразити, в яких рамках може змінюватись ентропія об'єднання двох інформаційних сигналів – X та Y. Вираз для умовної ентропії буде мати вигляд:

$$S(X|Y) = -\sum_{i=1}^{n}\ \sum_{j=1}^{n} p(y_i)\,p(x_i|y_j)\cdot\log_2 p(x_i|y_j). \quad (2.36)$$

А для незалежних рандомних випадків вірогідності дорівнюють безумовним. Припустимо, що складові X та Y сигнала *(X,Y)* незалежні, то вираз (2.14) можлива заміна $p(x_i|y_j)$ на $p(x_i)$, і тоді вона може бути перетворена до виду:

$$S(X|Y) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)\sum_{j=1}^{n} p(y_j). \quad (2.37)$$

Враховуючи, що $\sum_{j=1}^{n} p(y_j) = 1$ маємо вираз

$$S(X|Y) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i) = S(X). \quad (2.38)$$

Іншими словами, умовна ентропія при незалежних сигналах рівна безумовній ентропії, що призводить до $S(X,Y) = S(X) + S(Y)$.

У випадку коли інформаційні сигнали X, Y повністю залежні, значення ентропії системи будуть визначатися наступним чином.

Нехай сигнал X прийняв значення x_l, а сигнал Y значення y_k. У цьому випадку вірогідність $p(y_k|x_l) = 1$, а усі інші приймають нульове значення. Тоді у формулі (2.15) будуть надходити складові виду $p(x_i) \cdot 1 \cdot \log_2 1$, або $p(x_i) \cdot 0 \cdot \log_2 0$. У будь-якому випадку ці складові дорівнюють нулю, тому $S(X|Y) = 0$.

2.4 Алгоритмічне забезпечення автоматизованої системи оповіщення на виробництві

Для розрахунку ефективності функціонування, оперативності зв'язку та середнього часу очікування в мережі передачі інформації було розроблено алгоритм розрахунку. Комплексна система оповіщення на виробництві (СОВ) є територіально розподіленою обчислювальною мережею.

У разі виникнення нештатної ситуації на виробництві сигнал від датчиків надходить на автоматичні установки пожежної сигналізації (АУПС), встановлені безпосередньо на контрольованому об'єкті та підключені до СОВ.

Подія являє собою структуровану інформацію про позаштатну ситуації. Всі повідомлення, отримані від автоматичних систем пожежної сигналізації на об'єктах, що зберігаються на керуючому сервері. Події та повідомлення, що надходять до СОВ, можуть мати власний або присвоюваний СОВ рівень критичності.

Підтримувані СОВ рівні критичності:

– критичний – пожежа на об'єкті, позаштатна ситуація;

– важливий – зникнення зв'язку з автоматичною системою пожежної сигналізації;

– інформаційний – проведення технічного обслуговування автоматичної системи пожежної сигналізації та пожежогасіння на об'єкті.

На сервер надходить інформація від АУПС (рис. 2.5).

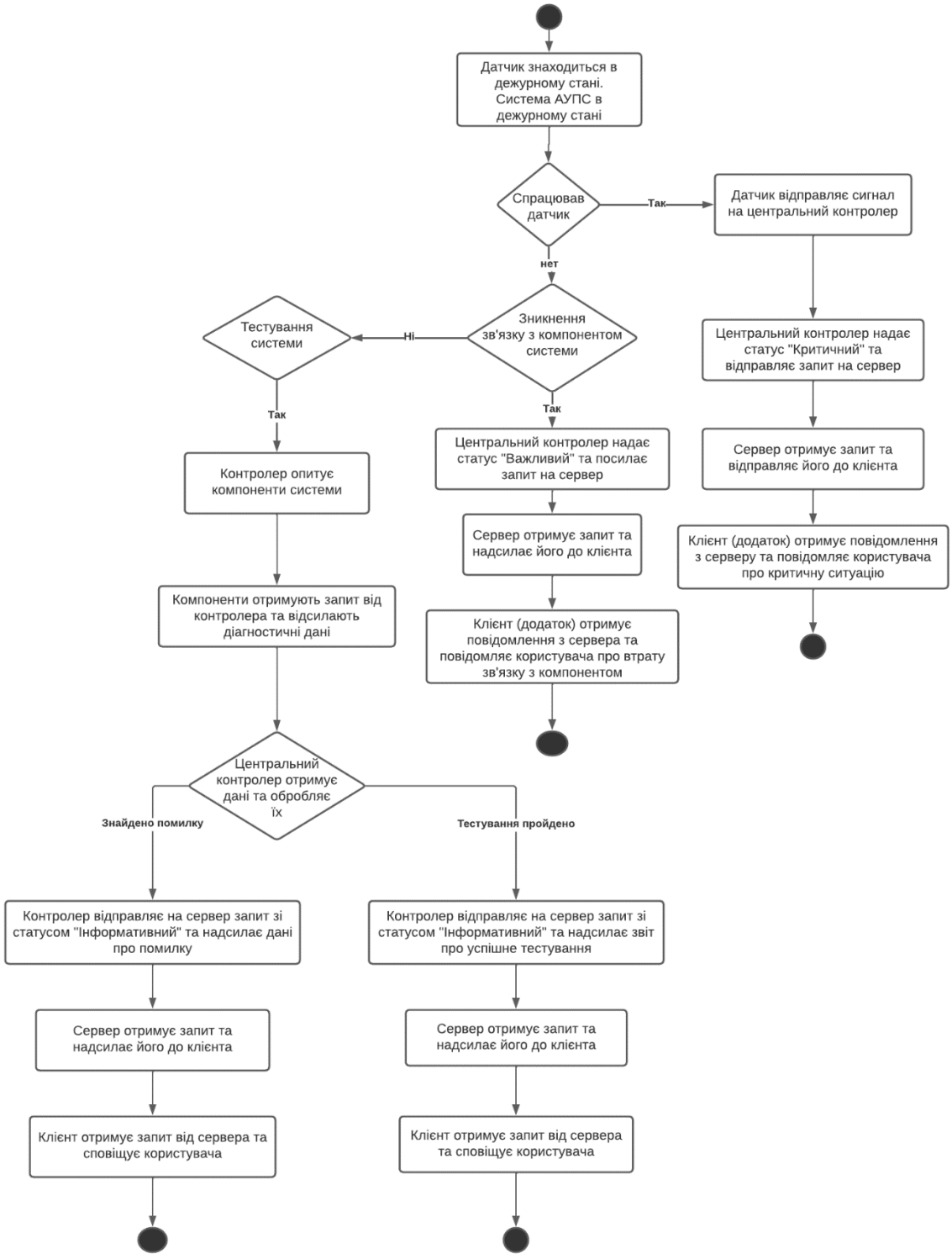


Рисунок 2.5 – Алгоритм роботи автоматизованої системи оповіщення

2.5 Висновки до другого розділу

У результаті другого розділу було побудовано алгоритм розрахунку, який полягає в тому, що визначення усіх можливих значень величин E, Q, $\overline{t_{оч}}$ ведеться у три етапи. На першому етапі визначається залежність E, Q, $\overline{t_{оч}}$ від N для різних значень λ та $\overline{T_{П}}$. Потім обчислюється залежність E, Q, $\overline{t_{оч}}$ від λ для конкретного значення N та різних значень $\overline{T_{П}}$. На третьому етапі визначається залежність величин E, Q, $\overline{t_{оч}}$ від для конкретного значення N та різних значень λ.

Також в результаті проведеного моделювання автоматизованої системи оповіщення на виробництві було отримано вираз (2.14), який дозволяє отримати величину інформаційної ентропії для оцінки надійності функціонування каналу передачі інформації.

Підвищення стійкості системи може бути досягнуто за рахунок створення додаткових каналів передачі інформації, які не працюють у нормальному стані. Проте якщо виникає сигнал тривоги, який йде від датчиків надлишкового тиску або інших, данні канали будуть задіяні, що не дасть можливості системі відмовити в умовах надзвичайної ситуації, обумовленої спалахами на підприємстві.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ СПОВІЩЕННЯ

3. 1 Аналіз існуючих фреймворків

При проектуванні мобільного додатку перш за все було потрібно обрати фреймворк. Спочатку було виділено Flutter та React Native. Обидва фреймворки використовуються для розробки мобільних програм. В цілому, у React Native більша власна бібліотека UI – елементів, ніж бібліотека віджетів Flutter [11-12].

Однак перевага останнього в даному випадку в тому, що він не настільки залежить від сторонніх бібліотек елементів, як React Native. Деякі елементи у них виявляють несумісність із конкретними платформами. Можна сказати, що Flutter в даному випадку більш універсальний і широко застосовується. Крім того, Flutter перевершує React Native і за продуктивністю, використовуючи цілком відмінний підхід до рендерингу. Так, Flutter створює власні віджети та використовує графічний процесор для рендерингу, а не запозичує нативні компоненти з інших платформ.

Написаний на мові Dart код Flutter компілює безпосередньо в оброблюваний процесором код ARM. Завдяки цьому програми, що створюються на Flutter, працюють помітно швидше. Тоді як React Native міст Javascript, який використовується для інтерпретації UI – елементів і викликає Java API або Objective-C для відображення відповідно компонентів iOS і Android, може уповільнювати роботу додатків.

Саме через показники швидкості та продуктивності був обран фреймвор Flutter.

Мобільний додаток буде розроблен на мові Dart – це мова програмування, яку розробляє компанія Google, позиціонуючи як мову структурованого програмування для Веб. Розробники вважали, що в

довгостроковій перспективі Dart може стати прогресивною заміною JavaScript, котрий потерпає від наявних в даний час проблем з розширюваністю, продуктивністю і підтримкою розробки складних застосунків. Мова має схожий на Java синтаксис, не вимагає явного визначення типів і її можна використовувати для створення серверних та клієнтських застосунків [20].

Мова має схожий на Java синтаксис, не вимагає явного визначення типів і може використовуватися для створення серверних і клієнтських застосунків. Для запуску всередині браузера код мовою Dart може бути перетворений в JavaScript-подання або запущений безпосередньо під управлінням спеціального JavaScript-інтерпретатора Dartboard. Підтримується вбудовування коду мовою Dart в HTML-сторінки, використовуючи MIME тип "application/dart". На стороні сервера застосунок на мові Dart може бути виконаний всередині спеціальної віртуальної машини, яка забезпечує продуктивність виконання близьку до компільованих в машинний код мов. Віртуальну машину Dart планують інтегрувати в майбутні версії браузера Chrome, що дозволить виконувати застосунки мовою Dart без компіляції в JavaScript. Мова підходить як для розробки одним програмістом невеликих скриптів без жорсткої структури, так і для створення високо масштабованих великих модульних проектів, підтримуваних великим колективом з потребою більш явної типізації для того, щоб уникнути плутанини і помилок. Код Dart завжди виконується тільки в рамках одної потоку, для організації паралельного виконання пропонується використовувати класи з атрибутом isolate. У кожному скрипті використовується власний простір імен, для використання зовнішніх об'єктів, функцій або змінних слід їх явно імпортувати за допомогою конструкції "import". Всі змінні, початково, діють тільки в межах поточного скрипту і не експортуються глобально [13].

Dart має такі особливості як забезпечення швидкого запуску і високої продуктивності для всіх сучасних веб-браузерів і різних типів оточень, від

портативних пристроїв до потужних серверів. Можливість визначення класів і інтерфейсів, що дозволяють використовувати інкапсуляцію і повторно використовувати існуючі методи і дані. Необов'язкове вказування типів, використовувати чи ні статичні типи вирішує розробник. Вказування типів дозволяє спростити зневадження і виявлення помилок, робить код яснішим і читаним, спрощує його доопрацювання та аналіз сторонніми розробниками. Серед підтримуваних типів: різні види хешів, масивів і списків, черги, числові і рядкові типи, типи для визначення дати і часу, регулярні вирази (RegExp). Можливо створення своїх типів. Для організації паралельного виконання пропонується використовувати класи з атрибутом isolate, код яких виконується повністю в ізольованому просторі в окремій області пам'яті, взаємодіючи з основним процесом через відправку повідомлень. Підтримка використання бібліотек, що спрощують підтримку і зневадження великих веб-проектів. Сторонні реалізації функцій можуть підключатися у вигляді поділюваних бібліотек. Застосунки можна розбити на частини і доручити розробку кожної з частин окремій команді програмістів. Набір готових інструментів для підтримки розробки мовою Dart, включаючи реалізацію засобів динамічної розробки та зневадження з виправленням коду на льоту ("edit-and-continue"). Можливість створювати однорідні системи, що охоплюють як клієнтську, так і серверну частину. Використання однієї мови та інструментарію для клієнтських і серверних компонентів спрощує процес кодування і позбавляє від постійної зміни контексту [21-22].

3.2 Платформа розробки

При розробці додатку на фреймворку Flutter було обрано платформу розробки Firebase [14-15].

На сьогодняшній це найбільш перспективне поєднання, оскільки використання Firebase дозволяє уникнути етапу створення серверного коду. Firebase Realtime Database надає бек-енд як службу в режимі реального часу

для створення мобільних програм, включаючи автентифікацію, зберігання, хостинг та базу даних. Це дозволяє розробникам API синхронізувати дані програми між різними клієнтами та зберігати їх у хмарному сервісі Firebase, не створюючи власний сервер. Все це значно прискорює процедуру створення мобільного кроссплатформенного додатка без втрати якості.

3.3 Аналіз архітуктурних патернів

У Flutter визначається не архітектурний патерн, як такий, з подальшою побудовою програми за правилами обраного рішення, а State Management System – так звану архітектуру управління станом. Прийшло це з галузі web-розробки. І ось тут виникає наступний вибір:

– Vanilla/Native state;

– Provider/Scoped Model;

– Business Logic Component (BLoC);

– Redux;

– GetX.

Vanilla/Native state (рис. 3.1), іноді кажуть, що найкраща архітектура – це відсутність архітектури. Правило написання коду згідно цього патерну викладено у документації до фреймворку Flutter [23].

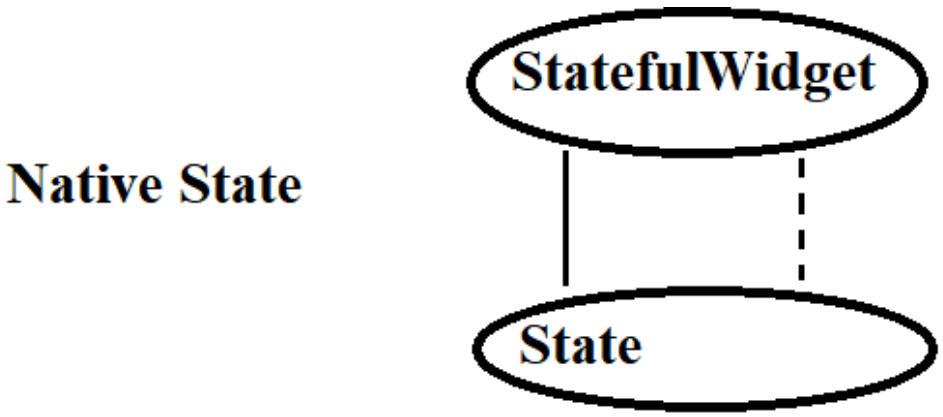


Рисунок 3.1 – Native State

Особливості такого підходу – висока швидкість розробки, а також відсутність поділу між бізнес-логікою та User Interface (UI). Але на практиці не треба застосувати такий підхід при розробці великої програми, інакше можливі проблеми з налагодженням та подальшим перевикористанням коду.

У Provider/Scoped Model (рис. 3.2), бізнес-логіка винесена з UI. Загалом все дуже просто, плюс є можливість перевикористання коду. Бібліотека Flutter Provider є сумішшю Ін'єкції Залежностей (Dependency Injection) і управління станом, побудована за допомогою віджетів і для віджетів [16-18].

Він використовує Флаттер-віджети для управління станом замість Dart класів, таких як Steam. Причина в тому, що віджети дуже прості, але надійні та масштабовані [24-25].

Використовуючи віджети для керування станом, провайдер може гарантувати:

– зручне керування кодом, завдяки примусовому односпрямованому потоку даних;

– тестованість і компонованість, тому що завжди можна перевизначати значення;

– надійність, тому що важко забути обробити сценарій оновлення моделі або віджету.

Однак проблеми все ж таки можливі, і починаються вони при роботі з великими і навіть із середніми проектами.

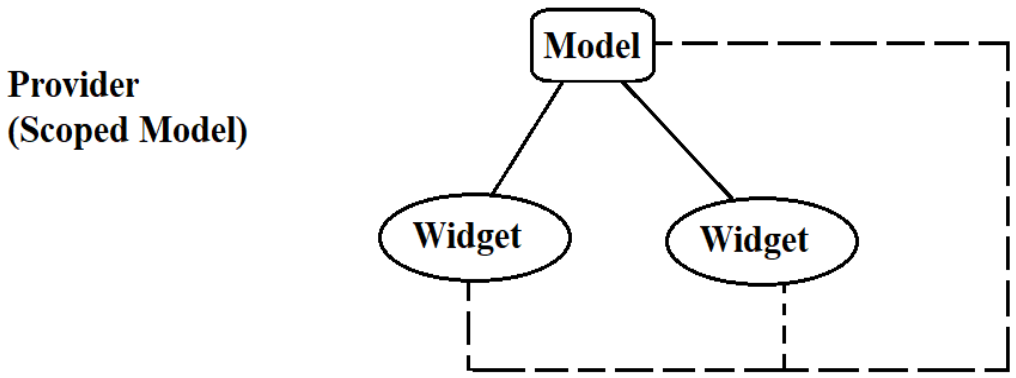


Рисунок 3.2 – Provider/Scoped Model

BLoC (рис. 3.3) – це акронім від "Business Logic Component" (компонент бізнес-логіки). Як випливає з назви, це клас, що відокремлює бізнес-логіку програми від інтерфейсу користувача. Такий компонент містить код, який можна повторно використовувати будь-де: в іншому модулі, на іншій платформі, в іншому додатку [19].

Відділення UI від бізнес-логіки для Flutter життєво потрібне. Оскільки складно шукати по дереву віджетів (UI – компонентів) потрібну логіку. Особливо якщо верстка і так містить дуже багато коду і розташована у різних файлах. Крім того, згідно Clean Architecture, у UI не повинно бути нічого зайвого. Так само як і бізнес-логіка нічого не повинна знати про UI, який до неї звертається. Завдяки такому ретельному поділу відповідальності отримується повністю ізольований компонент, який можна легко тестувати незалежно від UI та використовувати в іншому оточенні. Цей компонент і є Bloc.

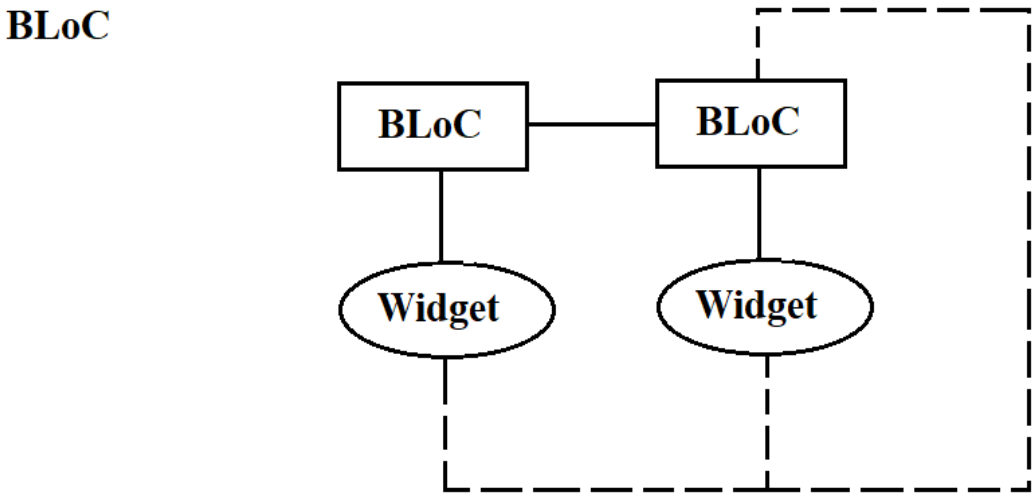


Рисунок 3.3 – BLoC

У патерні BLoC можна виділити чотири основні рівні додатків:

– UI – Користувальницький інтерфейс. Він містить всі компоненти програми, які видно користувачеві та з якими можна взаємодіяти. Оскільки у Flutter всі частини інтерфейсу користувача є віджетами, можна сказати, що всі вони належать цьому рівню (класу);

– BLoC – це класи, які виступають в якості сполучного середовища між даними та компонентами інтерфейсу користувача. BLoC приймає передані від нього події, а після отримання відповіді видає відповідний стан;

– Repository.(Сховище) відповідає за отримання частини інформації з одного або декількох джерел даних та її обробку для класів інтерфейсу користувача [20];

– Data sources (Джерела даних) – це класи, які надають дані додатку з усіх джерел даних, включаючи базу даних(database), мережу(network),shared preferences тощо.

BLoC спирається на два основні компоненти:

– Events (Події), що передаються з інтерфейсу користувача, які містять інформацію про конкретну дію, яка повинна бути оброблена BLoC;

– States (Стани), які показують, як інтерфейс користувача повинен реагувати на зміну даних. Кожен BLoC має свій початковий стан, що визначається під час створення.

Redux. Однією з відмінних рис цього патерну є те, що він повністю базується на реактивності. Реактивне програмування – це програмування з асинхронними потоками даних. У традиційному імперативному стилі зазвичай пишуть код наступного змісту: текстове поле, призначається йому обробник, що реагує на введення нового символу, додається об'єкт, у якого викликається метод оновлення текстового поля в моделі. Чітко описується кожен крок. У реактивному підході все виглядає трохи інакше. Текстове поле зв'язується з конкретною змінною моделі. І як тільки користувач починає щось друкувати, ця змінна відразу набуває того значення, яке зараз міститься в текстовому полі.

Redux (рис. 3.4) – це архітектура, спочатку створена для JavaScript і використовується в додатках, які створені з використанням reactive frameworks (таких як React Native або Flutter). Redux – це спрощена версія архітектури Flux, створена Facebook. По суті цей патерн має три головні пункти:

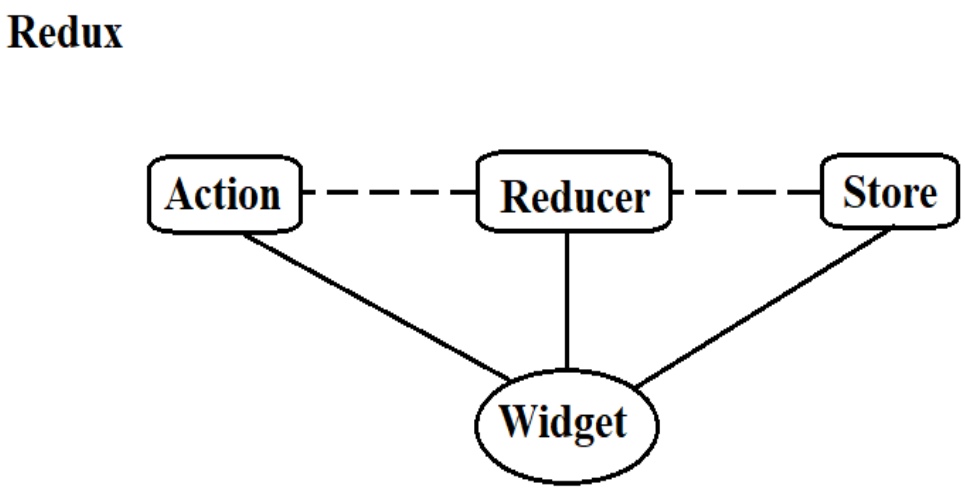


Рисунок 3.4 – Redux

– єдине джерело правди / single source of truth – весь state додатку зберігається тільки в одному місці (називається store);

– стан доступний тільки для читання/state is read-only – для зміни state програми необхідно відправити actions(дія), після чого створиться новий state;

– зміни виконуються за допомогою чистих функцій/pure functions — чиста функція (для простоти, це функція без side effects) приймає поточний state додаток та action та повертає новий state додаток.

Особливості – більш жорсткі обмеження та менше свободи вибору. Але навіть за рахунок цього вдається досягти властивої для Redux гнучкості і простоти налагодження. Проте у цього підходу виділяють такий суттєвий мінус як багатошаровість.

GetX має надзвичайно легкий і простий склад менеджера, який не використовує ChangeNotifier, задовольняє вимоги.

GetBuilder націлений саме на контроль кількох станів. Наприклад, якщо додали 30 продуктів у корзину, далі було видалено один, одночасно з цим оновлюється список, оновлюється ціна, а значок у корзині покупок оновлюється до меншого числа. Такий підхід робить GetBuilder вбивчим, тому що він групує стани та змінює їх усе відразу без будь-якої "визначної логіки" для цього. GetBuilder був створений з урахуванням такого роду ситуацій, тому що для тимчасової зміни стану можливо використовувати setState, і для цього не знадобиться менеджер складається.

Таким чином, якщо потрібен окремий контролер, можливо призначити для нього ідентифікатори або використовувати GetX. Але важливо пам’ятати, що чим більше «індивідуальних» виджетів, тим більше ресурсів буде завантажено GetX, у той час як продуктивність GetBuilder повинна бути вищею при багаторазових змінах стану.

GetX дійсно зручний, лаконічен, функціональний, виразний. Але іноді його функціоналу не достатньо.

MobX (рис. 3.5) це просте, випробуване в комерційній роботі рішення для керування станом програми. MobX це автономна бібліотека, що дозволяє зробити управління станом простим, повернувшись до кореня проблеми: він унеможливлює інконсистентність стану. Стратегія досягнення цього досить проста: переконається, що все, що може бути вийнято зі стану, буде вийнято. Автоматично.

Концептуально MobX обробляє програму як електронна таблиця.

По-перше, є стан State програми. Графи об'єктів, масивів, примітивів, посилань, які формують модель програми.

По-друге, є похідні Derivations. Зазвичай це будь-яке значення, яке може бути обчислено автоматично з даних стану програми.

Реакції Reactions дуже схожі на похідні Derivations. Основна відмінність: вони не повертають значення, але запускаються автоматично,

щоб виконати якусь роботу. Зазвичай це з I/O. Вони перевіряють, що DOM оновився або запити мережі виконалися вчасно.

Зрештою, є дії Actions. Дії це всі ті штуки, які змінюють стан. MobX простежить, щоб усі зміни у стані програми, викликані діями, автоматично обробилися всіма похідними та реакціями. Синхронно та без перешкод.

Єдиний реальний мінус MobX – він дає вам занадто багато свободи у тому, як структурувати код, зберігати та обробляти дані. Оскільки це суттєво ускладнює процесс проектування застосунку.

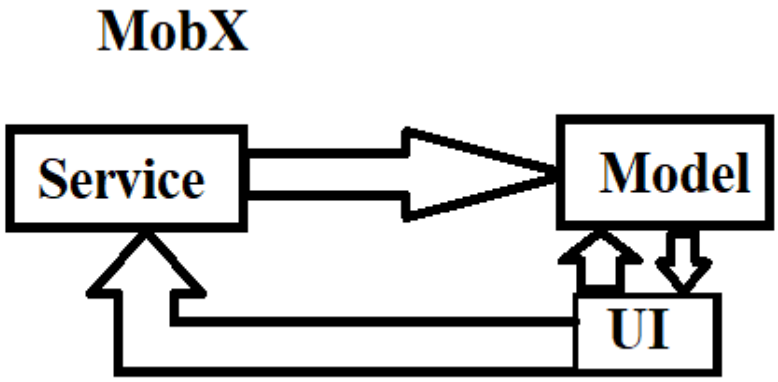


Рисунок 3.5 – MobX

3.4 Розроблення програмного коду

При розробці мобільного додатку перш за все було прийнято рішення по вибору необхідних бібліотек:

– firebase_core – це плагін Flutter для використання Firebase Core API, який дозволяє підключатися до головних програм Firebase;

– firebase_messaging – це плагін Flutter для використання API Firebase Cloud Messaging;

– flutter_local_notifications – це міжплатформний плагін для відображення локальних сповіщень;

– overlay_support – цей пакет допомогає накладання, полегшує створення тостів і сповіщень у програмі;

– flutter_bloc – пакет надає доступ до віджетів Flutter, які спрощують реалізацію шаблону проектування BLoC (компонент бізнес-логіки). Створено для використання з пакетом управління станом блоку;

– firebase_analytics – плагін Flutter для Google Analytics Firebase, рішення для вимірювання додатків, яке надає уявлення про використання додатків і залучення користувачів на Android та iOS;

– material – плагін Flutter для використання віджетів Flutter які працюють з material дизайном.

Так як додаток має архітектуру BLoC, то було створено клас NotificationBloc який відповідає за бізнес логіку проекту та оновлення стейту. За для опису цього самого стейту було створено клас NotificationState від якого наслідуються три класи GotNotificationState, GotNotificationErrorState, InitialState.

Точкою входа у додатку є функція void main (рис. 3.6) яка у свою чергу повертає клас MyApp який наслідуїться від StatelessWidget (рис. 3.7).

```
Run | Debug | Profile
void main() {
  runApp(const MyApp());
}
```

Рисунок 3.6 – Точка входу у додатку

```dart
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return BlocProvider<NotificationBloc>(
      create: (context) => NotificationBloc()..registerNotification(),
      child: OverlaySupport(
        child: MaterialApp(
          debugShowCheckedModeBanner: false,
          title: 'System Notification',
          theme: ThemeData(
            primarySwatch: Colors.blue,
          ), // ThemeData
          home: HomeScreen(),
        ), // MaterialApp
      ), // OverlaySupport
    ); // BlocProvider
  }
}
```

Рисунок 3.7 – Клас MyApp

У данному классі є віджет BlocProvider, який робить класс NotificationBloc доступним для дерева віджетів, яке передається в параметрі child. У данному випадку це MaterialApp – зручний віджет, який містить кілька віджетів, які зазвичай потрібні для програм Material Design. Він будується на WidgetsApp, додаючи спеціальні функції матеріального дизайну, такі як AnimatedTheme і GridPaper.

MaterialApp у своєму параметрі home звертається до класу HomeScreen (рис. 3.8), який наслідуїться від StatefullWidget.

У даному класі є BlocBuilder, він створює віджет у відповідь на нові стани. BlocBuilder є аналогом StreamBuilder, але має спрощений API, щоб зменшити кількість необхідного шаблонного коду, а також покращення продуктивності для окремих блоків. Будь ласка, зверніться до BlocListener, якщо ви хочете "зробити" що-небудь у відповідь на зміни стану, такі як навігація, показ діалогового вікна тощо. Якщо параметр bloc опущено, BlocBuilder автоматично виконає пошук за допомогою BlocProvider і поточного BuildContext.

```dart
class HomeScreen extends StatefulWidget {
  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Система оповіщення'),
      ), // AppBar
      body: BlocBuilder<NotificationBloc, NotificationState>(
        builder: (context, state) {
          if (state is GotNotificationState) {
            return notificationScreen(state.notificationInfo);
          } else if (state is GotNotificationErrorState) {
            return errorScreen();
          }
          return defaultScreen();
        },
      ), // BlocBuilder
    ); // Scaffold
  }
```

Рисунок 3.8 – Клас HomeScreen

Якщо у якості стейту повертається GotNotificationState то BlocBuilder повертає notificationScreen (рис. 3.9).

```dart
Widget notificationScreen(PushNotification notification) {
  return Center(
      child: Column(
    crossAxisAlignment: CrossAxisAlignment.center,
    children: [
      const SizedBox(height: 100),
      Text(
        'Увага ви отримали повідомлення о ${notification.sentTime}',
        style: const TextStyle(
          fontWeight: FontWeight.bold,
          fontSize: 22.0,
        ), // TextStyle
      ), // Text
      const SizedBox(height: 50),
      Text(
        notification.body.toString(),
        style: const TextStyle(
          fontWeight: FontWeight.bold,
          fontSize: 16.0,
        ), // TextStyle
      ), // Text
    ],
  )); // Column // Center
}
```

Рисунок 3.9 – Widget notificationScreen

Якщо у якості стейту повертається GotNotificationErrorState то BlocBuilder повертає errorScreen (рис. 3.10).

```
Widget errorScreen() {
  return const Center(
    child: Text(
      'Будь ласка дайте згоду на отримання повідомлення',
      textAlign: TextAlign.center,
      style: TextStyle(
        color: Colors.black,
        fontSize: 20,
      ), // TextStyle
    ), // Text
  ); // Center
}
```

Рисунок 3.10 – Widget errorScreen

Якщо у якості стейту не виконується ніяка умова, то BlocBuilder повертає defaultScreen (рис. 3.11).

```
Widget defaultScreen() {
  return const Center(
    child: Text(
      'Гарного дня!',
      textAlign: TextAlign.center,
      style: TextStyle(
        color: Colors.black,
        fontSize: 20,
      ), // TextStyle
    ), // Text
  ); // Center
}
```

Рисунок 3.11 – Widget defaultScreen

У класі NotificationBloc описана функція registerNotification (рис. 3.12). Вона підключається до Firebase та прослуховує сповіщення. Також виявляє чи надав користувач дозволення на отримання повідомлення.

```dart
void registerNotification() async {
  late final FirebaseMessaging _messaging;
  // 1. Initialize the Firebase app
  await Firebase.initializeApp();

  // 2. Instantiate Firebase Messaging
  _messaging = FirebaseMessaging.instance;

  // 3. On iOS, this helps to take the user permissions
  NotificationSettings settings = await _messaging.requestPermission(
      alert: true,
      badge: true,
      provisional: false,
      sound: true,
      criticalAlert: true);

  if (settings.authorizationStatus == AuthorizationStatus.authorized) {
    print('User granted permission');

    // For handling the received notifications
    FirebaseMessaging.onMessage.listen((RemoteMessage message) {
      // Parse the message received
      String sentTime = message.sentTime.toString();
      String sentTimeWithoutMiliseconds =
          sentTime.substring(0, sentTime.toString().length - 4);
      PushNotification notification = PushNotification(
        title: message.notification?.title,
        body: message.notification?.body,
        sentTime: sentTimeWithoutMiliseconds,
      );

      if (notification != null) {
        // For displaying the notification as an overlay
        emit(GotNotificationState(notification));
      }
    });
  } else {
    print('User declined or has not accepted permission');
  }
}
```

Рисунок 3.12 – Функція registerNotification

3.5 Тестування роботи застосунку

Спочатку на телефон з операційною системою Android було завантажено застосунку. Ніяких додаткових налаштувань це не потребує.

У якості початкової сторінки користувач бачe перший екран (рис. 3.13).

Рисунок 3.13 – Початкова сторінка

Якщо система для супроводження автоматизованих систем оповіщення на виробництві знаходиться у стані моніторингу, то результат діагностування вона надсилає на сервер. Після чого вже Firebase надсилає повідомлення о стані системи (рис. 3.14 – 3.15).

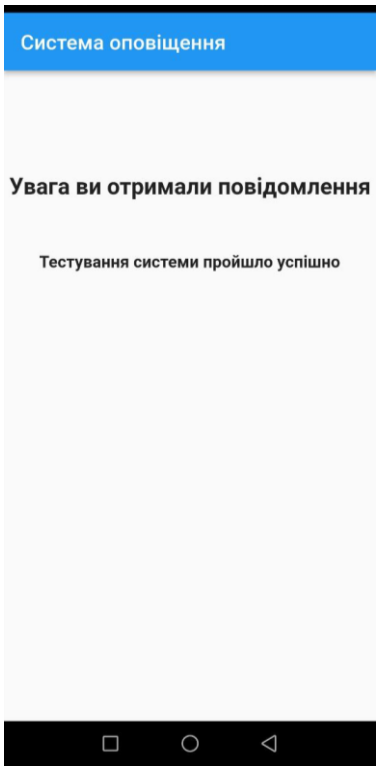


Рисунок 3.14 – Моніторинг системи пройшов успішно

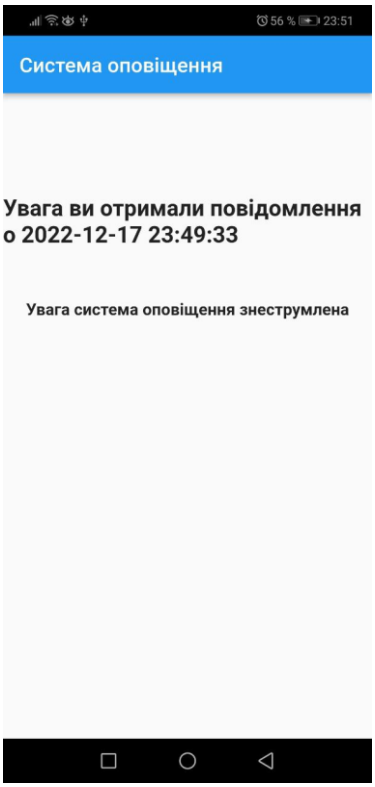


Рисунок 3.15 – Моніторинг системи виявив помилку

У черезвичайній ситуації, коли спрацьовує один чи декілька датчиків, центральний контролер відправляє запит на сервер та користувач отримує наступне повідомлення (рис. 3.16).

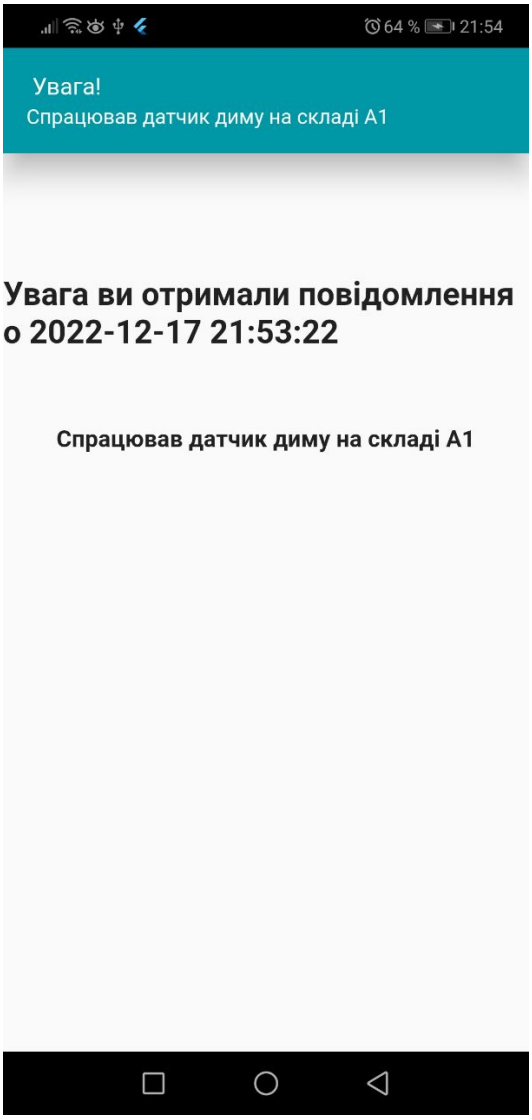


Рисунок 3.16 – Спрацював датчик диму

Навіть якщо на момент надходження повідомлення застосунок знаходиться у фоновому режимі користувач все одно отримає повідомлення (рис. 3.17-3.18).

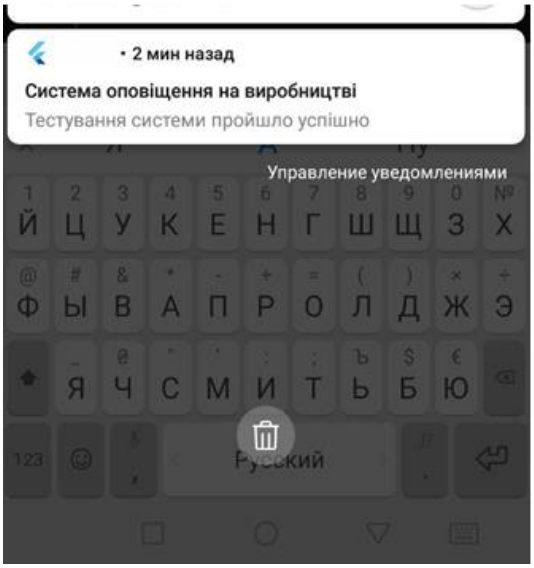


Рисунок 3.17 – Повідомлення після тестування у фоновому режимі

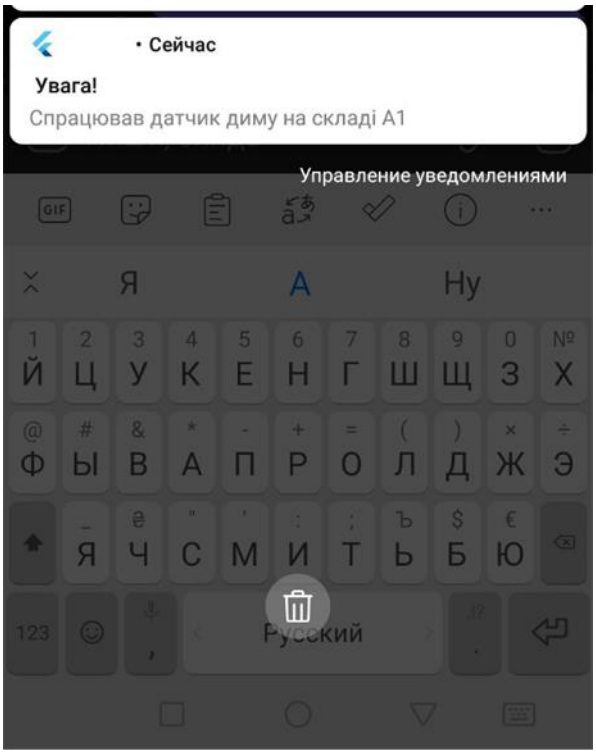


Рисунок 3.18 – Критичне повідомлення у фоновому режимі

3.6 Висновки до третього розділу

Було проведено аналіз існуючих фреймворків та обрано Flutter у якості платформи розробки було виділено серед інших доступних Firebase. Також

було досліджено архітектурні патерни Flutter та підібрано найбільш відповідний до системи.

У результаті проведених досліджень було розроблено мобільний додаток та протестовано його роботу.

4 ОХОРОНА ПРАЦІ

4.1 Фактори, що впливають на здоров’я працівників виробництва

Охорона праці є важливою системою заходів, дотримання яких дозволяє зберегти як життя, і здоров'я працівників під час виконання ними своїх обов'язків.

Охорона праці на виробництві складається з юридичних і правил безпеки, частина з яких диктується державою, а частина розробляється безпосередньо на підприємстві. На здоров'я людей на виробничому приміщенні можуть впливати різні шкідливі та небезпечні фактори.

Найпоширеніший небезпечний фактор – електричний струм. Він небезпечний своєю всюдисущістю та неможливістю його ідентифікації до моменту отримання травми [26].

Важливе питання освітленості робочого місця. Особливо – для персоналу, який проводить більшу частину свого робочого часу біля комп'ютера.

Не менш важливим питанням для забезпечення безпеки в офісі є дотримання правил пожежної безпеки. Потрібно не лише навчити персонал грамотно поводитися з побутовою та офісною електротехнікою, а й організувати тренінги з протипожежної евакуації із включенням систем оповіщення про початок пожежі. Маючи всі ці знання і навички, можна за короткий термін виявити джерело пожежі і перешкодити його подальшому поширенню.

Евакуаційні тренінги, досконале знання шляхів для аварійного виходу у непередбачених складних ситуаціях – запорука протипожежної безпеки офісу. Найбільший відсоток смертей при пожежі пов'язаний з панікою та отруєнням чадним газом через неможливість знайти вихід із офісу.

4.2 Освітленість на виробництві

Світло є природною умовою життєдіяльності людини. Він надає позитивний вплив на емоційний стан людини, впливає на обмін речовин, серцево-судинну, нервово-психічну системи, є важливим стимулятором не лише зорового аналізатора, а й організму загалом. Понад 80% всієї інформації про довкілля надходить у мозок людини через очі.

Видиме випромінювання (світло) – ділянка загального електромагнітного спектру, що складається із 7 основних кольорів, саме він безпосередньо викликає зорове відчуття. Видимі випромінювання зазвичай вимірюють у нанометрах (1 нм = 1×10^{-9} м). Чутливість ока максимальна у зеленій області діапазону при довжині хвилі l = 554 нм.

Раціональне освітлення виробничих приміщень надає позитивний психофізіологічний вплив на працюючих, сприяє підвищення продуктивності праці, забезпечення її безпеки, збереження високої працездатності людини у процесі праці. Оптимізація виробничого освітлення сприяє підвищенню продуктивність праці на 10 – 20 %, зменшення шлюбу на 20 % і зниження кількості нещасних випадків на 30 %.

При недостатній освітленості та поганій якості освітлення стан зорових функцій є незадовільний, у процесі виконання роботи підвищується втома очей, зростає небезпека травматизму. Встановлено, що погане освітлення є причиною приблизно 5% нещасних випадків на підприємствах, а також очних хвороб, головного болю, швидкої стомлюваності.

З іншого боку, існує небезпека негативного впливу на органи зору надто великої яскравості (блискучості) джерел світла. Наслідком цього може стати тимчасове порушення зорових функцій ока (явлення сліпимості).

З метою забезпечення нормальних умов праці та захисту зору людини у виробничих приміщеннях має бути встановлене освітлення, яке відповідає вимогам відповідних і правил [27].

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було проведено дослідження, що описує проблематику використання системи мовного сповіщення у якості системи оповіщення на виробництві. Під час досліження було вирішено ці питання через розробку мобільного додатку на фреймворку Flutter мові Dart з використанням Firebase у якості платформи для розробки, завдяки використанню цих технологій додаток буде доступний для телефонів на операційних системах Android та IOS. Саме така система вирішить питання недоступності голосового сповіщення за для людей з вадами слуху через світлове повідомлення о екстреній ситуації.

Було побудовано алгоритмічне забезпечення автоматизованої системи оповіщення на виробництві та її математичну модель. Розраховано стійкість автоматизованої системи. Спроектовано програмне забезпечення, яке базується на дослідженні існуючих фреймворків, платформ розробки та архітектурних патернів.

За результатами досліджень кваліфікаційної роботи було підготовлено доповіді на: VI-ої Міжнародної VI конференц Виробництво & Мехатронні Системи (м. Харків, 21-22 жовтня 2021 р.) [24]; Матеріали V-ої Міжнародної V конференц Виробництво & Мехатронні Системи (м. Харків, 21-22 жовтня 2022 р.).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шостенко С.С. Архітектура програмного забезпечення для супроводження автоматизованих систем оповіщення на виробництві / С.С. Шостенко, О. О. Чала // Матеріали VI-ої Міжнародної VI конференц Виробництво & Мехатронні Системи Тези доповідей. − 2022: іі, Харків, ХНУРЕ, 21-22 жовтня 2022. С. 115-117.

2. Шостенко С.С, Буць Д. Розроблення програмно-організаційного забезпечення для супроводження автоматизованих систем оповіщення на виробництві/ Шостенко С.С, Буць Д.// Матеріали V-ої Міжнародної V конференц Виробництво & Мехатронні Системи Тези доповідей. − 2021: іі, Харків, ХНУРЕ, 21-22 жовтня 2021. С. 95-100.

3. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. Структура та правила оформлення. – Введ. 2015-06-22. – К. Держстандарт України, 2017 – 29 с.

4. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп’ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами», «Комп’ютерно-інтегровані технологічні процеси і виробництва», «Комп’ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. – Харків: ХНУРЕ, 2021. – 55 с.

5. Стандарт вищої освіти магістра за спеціальністю 151 «Автоматизація та комп’ютерно-інтегровані технології» галузі знань 15 «Автоматизація та приладобудування» затверджено і введено в дію Наказом Міністерства освіти і науки України від 10.08.2020 р. № 1022. Режим доступу : https://

mon.gov.ua/storage/app/media/vishchaosvita/zatverdzeni%20standarty/2020/08/10 / 151-avtomatizatsiya-ta-kitmagistr.pdf

6. Освітньо-професійна програма «Автоматизоване управління технологічними процесами». – Режим доступу : https://nure.ua/wpcontent/uploads/Education_programs/2021/2021_mag_151_opp_autp.pdf.

7. Освітньо-професійна програма «Комп'ютерно-інтегровані технологічні процеси і виробництва». – Режим доступу : https://nure.ua/wpcontent/uploads/Education_programs/2021/2021_mag_151_opp_kitpv.pdf.

8. Положення про роботу екзаменаційних комісій [Електронний ресурс] : Наказ ХНУРЕ від 09 лютого 2015 р. № 40. – Режим доступу : https://nure.ua/wpcontent/ uploads/Main_Docs_NURE/nakaz-ta-polozhennya-proporyadokstvorennya-ta-organizatsiyu-roboti-ekzamenatsiynih-komisiy....pdf.

9. Положення про академічну доброчесність [Електронний ресурс] : Наказ ХНУРЕ від 02 лютого 2021 р. № 50. – Режим доступу : https://nure.ua/wpcontent/ uploads/Main_Docs_NURE/polozhennja-pro-akademichnu-dobrochesnist.pdf.

10. Невлюдов І.Ш. Основи наукових досліджень: Навч. посібник / І.Ш. Невлюдов, Ю.М. Олександров, А.О. Андрусевич, О.О. Чала. Кривий Ріг: Криворізький коледж НАУ. 2019. 396 с. 35.

11. Невлюдов І.Ш. Дипломне проектування для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»: Навч. посібник / І.Ш. Невлюдов, А.О. Андрусевич, О.В. Токарєва, Г.В. Пономарьова. К. : пр. Космонавта Комарова, 1. 2016. 245 с. 14. Основи виробництва електронних апаратів / Невлюдов І.Ш. Харків: ТОВ «Компанія СМІТ». 2005. 598 с. 14. Невлюдов І.Ш. Виробничі процеси та обладнання об'єктів автоматизації: Підручник / Кривий Ріг: КК НАУ. 2017. 444 с.

12. Невлюдов І.Ш. Виробничі процеси та обладнання об’єктів автоматизації. Збірник задач: Навчальний посібник / І.Ш. Невлюдов, А.О. Андрусевич, Г.В. Пономарьова, А.О. Функендорф. Кривий Ріг: КК НАУ. 2018. 332 с.

13. Невлюдов І.Ш. Технічні засоби автоматизації: Підручник / І.Ш. Невлюдов, А.О. Андрусевич, О.І. Филипенко, Н.П. Демська, С.П. Новоселов. – Кривий Ріг : Криворізький коледж НАУ. 2019. 366 с.

14. Невлюдов І.Ш. Техніко-економічне обґрунтування інженерних рішень: Підручник / І.Ш. Невлюдов. Кривий Ріг : Криворізький коледж НАУ. 2019. 448 с.

15. Шостенко С. С. Дослідження мініатюрних лінійних п’єзоелектричних двигунів «Автоматизація та приладобудування» («Automation and Development of Electronic Devices» ADED-2019) [Електронний ресурс]: збірник студентських наукових статей / Харківський національний університет радіоелектроніки. – Харків : ХНУРЕ, 2020. – Вип. 2. – с. 11-14.

16. Шостенко С. С. Проектування п’єзоелектричного мікродвигуна «Автоматизація та приладобудування» («Automation and Development of Electronic Devices» ADED-2020)[Електронний ресурс]: збіірник студентських наукових статей / Харківський національний університет радіоелектроніки. – Харків : ХНУРЕ, 2020. – Вип. 1. – с. 171 – 177

17. Квєтний Р. Н. Комп’ютерне моделювання систем та процесів. Методи обчислень. Частина 1 : навчальний посібник / Богач І. В. та інші; за заг. ред. Р.Н. Квєтного. Вінниця: ВНТУ, 2017.193 с.

18. Гіль А. Промислові інтерфейси та протоколи передачі даних інтегрованих систем для автоматизованого управління в умовах Industry 4.0 / Гіль А., Чала О., Филипенко О. // Виробництво & Мехатронні Системи 2021: матеріали V-ої Міжнародної конференції, Харків, 21-22 жовтня 2021 р.: Харків, 2021. С.127-30.

19. Невлюдов І. Ш.Трансфер технологій у сучасній науці, освіті та виробництві в умовах четвертої промислової революції «ІНДУСТРІЯ 4.0» / Невлюдов І. Ш., Чала О. О., Олександров Ю. М. // Сучасний рух науки: тези доп. VIII міжнародної науковопрактичної інтернет-конференції, 3-4 жовтня 2019 р. Дніпро, 2019. Т.2 С.: 604-608.

20. Bortnikova, V., Nevliudov, I., Botsman, I., & Chala, O. (2019, June). Search Query Classification Using Machine Learning for Information Retrieval Systems in Intelligent Manufacturing. In ICTERI (pp. 460-465).

21. Підтримка життєвого циклу у виробничій інженерії: монографія / І. Ш. Невлюдов, О. І. Филипенко, А. О. Андрусевич, М.Г. Стародубцев. – Кривий Ріг: Криворізький коледж НАУ, 2019. – 252 с.

22. Yevsieiev, V. Program code automated system development at early stage of software lifecycle / V. Yevsieiev // Наукові праці Донецького національного технічного університету. Серія: «Обчислювальна техніка та автоматизація». – Покровськ: ДВНЗ «ДонНТУ». Випуск 1 (30). – 2017. – С. 69 – 78.

23. Saska, Martin & Mejia, Juan & Stipanovic, Dusan & Vonasek, Vojtech & Schilling, Klaus & Preucil, Libor. (2013). Control and navigation in manoeuvres of formations of unmanned mobile vehicles. European Journal of Control. 19. 157–171.DOI:10.1016/j.ejcon.2012.10.003.

24. Юрчак О., Степанець О., Некрашевич О. Виробничі КПЕ (ключові показники ефективності): актуальний стан та перспективи розвитку в Україні. Стандарт ISO 22400 – Київ: АППАУ, 2019 – 45 с.

25. Базилевич Д.С., Лісовол Є.В., Самсонкін В.М., Фельдман А.О., Юрчак О.В. Глосарій термінів напрямку „Індустрія 4.0“. – Київ, Інститут стратегічних досліджень ім.Голди Меїр, 2018. – 21 с.

26. Репін Ю. В. Безпека та захист людини в надзвичайниї ситуаціях: навч. посіб. Миколаїв, 2005. 192 с.

27. Єфремова О. С. Збірник інструкцій з охорони праці: навч.посіб. 2008. 384 с.