Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

Пояснювальна записка

до бакалаврської дипломної роботи на тему:

«Одномодульна мультисенсорна система безпеки»

Виконав: студент 2 курсу, групи 1КІ-20мс

спеціальності 123 — Комп'ютерна інженерія

Ярошевський М.М.

Керівник к.т.н., доцент кафедри ОТ

Томчук М. А.

«20» 06 2022 р.

Рецензент к.т.н., доцент кафедри ЗІ

Куперштейн Л. М.

«21» 06 2022 р.

Допущено до захисту

Зав. кафедри ОТ, д.т.н., проф.

Азаров О. Д.

«22» 06 2022 р.

Вінниця ВНТУ — 2022 рік

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень перший (бакалаврський)

Галузь знань — 12 — Інформаційні технології

Спеціальність — 123 – «Комп'ютерна інженерія»

Освітньо — професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ, д.т.н., проф.

Азаров О. Д.

"08" 02 2022року

З А В Д А Н Н Я

НА БАКАЛАВАРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Ярошевському Максиму Миколайовичу

1 Тема роботи «Одномодульна мультисенсорна система безпеки» керівник роботи Томчук Микола Антонович, затверджені наказом вищого навчального закладу від «24» 03 2022року №66

2 Термін подання студентом роботи «13» 06 2022 року.

3 Вихідні дані до роботи: технічний опис програмного застосунку, мова програмування C++ і PHP з використанням HTML, та вільної системи керування реляційними базами даних MySQL.

4 Зміст текстової частини (перелік питань, які потрібно розробити): вступ, аналіз сучасних методів розробки, і також перегляд сучасних аналогів і варіантний вибір засобів для розробки даної роботи на мові програмування C++ і PHP, програмна реалізація.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): структурна схема пристрою з базовими функціями, блок схема алгоритму роботи програми.

6 Консультанти розділів роботи наведені в таблиці 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Томчук М.А., к.т.н., доц.		

7 Дата видачі завдання «05» 02 2022 року.

8 Календарний план виконання приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломного роботи	Термін виконання		Примітка
		початок	закінчення	
1	Постановка задачі роботи	09.02.2022		вик.
2	Пошук матеріалів по технологіям розробки на мові C++ і PHP	10.02.2022	25.02.22	вик.
3	Аналіз та вибір технології для роботи із базою даних	26.02.22	28.02.22	вик.
4	Обгрунтування та вибір засобів реалізації структури схеми пристрою і алгоритму роботи програми.	29.02.22	06.03.22	вик.
5	Проектування програмного забеспечення для пристрою	07.03.22	25.04.22	вик.
6	Розробка програмного забеспення пристрою	26.04.22	20.05.2022	вик.
7	Оформлення пояснювальної записки та ілюстративного матеріалу	21.05.2022	04.06.2022	вик.
8	Перевірка якості виконання бакалаврської роботи та усунення недоліків	07.06.2022		вик.

Студент Ярошевський М.М.

Керівник роботи Томчук М.А.

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 84 сторінок формату А4, на яких є 31 рисунків, 8 таблиць, список використаних джерел містить 20 найменувань.

У бакалаврській дипломній роботі проведено детальний аналіз технології і методів для роботи системи безпеки і створенню пристрою мультисенсорної автономної сигналізації з багатоканальним оповіщенням та програмного забезпечення для управління її роботою.

У результаті проведення досліджень розроблено пристрій сигналізації з багатоканальним оповіщенням та програмне забезпечення для управління нею, що дозволяє здійснювати обмін інформацією про стан об’єктів з хмарним середовищем. Розробка може здійснювати оповіщення про стан апаратних датчиків в хмарне середовище, а також розсилати сповіщення про події з використанням різних каналів зв’язку.

Описано роботу розподіленої системи сигналізації з багатоканальним оповіщенням і роботу програмного забезпечення щоб управляти нею і використання хмарного середовища, оповіщення апаратних датчиків в хмарне середовище, також розсилати сповіщення про події.

Ключові слова: C++, PHP, MySQL, Arduino Nano.

ABSTRACT

The bachelor's thesis consists of 84 A4 pages, which contain 31 figures, 8 tables, the list of used sources contains 20 titles.

In the bachelor's thesis the detailed analysis of technology and methods for work of security system and creation of the device of the multisensor autonomous alarm system with the multichannel notification and the software for management of its work is carried out.

As a result of research, a multi channel alarm device and software for its management have been developed, which allows exchanging information about the state of objects with a cloud environment. The developer can notify the status of hardware sensors in the cloud environment, as well as send notifications about events using different communication channels.

Describes a multi channel alarm device and software to control it and use the cloud environment to alert hardware sensors to the cloud environment to also send event notifications.

Keywords: C ++, PHP, MySQL, Arduino Nano.

ЗМІСТ

ВСТУП

Сучасні системи беспеки бюджетних домашніх та офісних сигналізації досить часто позбавлені можливості підключення стандартних провідних датчиків. Більш дорогі системи в свою чергу потребують складного налаштування, яке можуть виконати лише спеціалісти даної галузі та є занадто дорогими для більшості населення. Зараз ринок потребує охоронні сигналізації для дому, які є недорогими і мультисенсорними. Надійність охоронної сигналізації є важливим пунктом охоронної системи. Нині на ринку не було виявлено рішень, які одночасно підтримують провідні датчики, багатоканальне оповіщення та інтеграцію з сучасними мобільними пристроями без використання платних комерційних підписок. Саме тому прийнято рішення створити прототип, який зможе розширити можливості бюджетних сигналізації без використання дорожчої компонентної бази. Для вивчення предметної області знайдено інші існуючі рішення сигналізації, щоб дослідити.

Об’єкт дослідження: принципи створення пристроїв на базі мікроконтролерів та мікропрограм для них, що дозволяють забезпечити обмін інформацією про стан об’єктів з хмарним середовищем.

Предмет дослідження: створення пристрою на базі Arduino Nano, мікропрограми для обміну інформацією про стан об’єктів, а також скриптів синхронізації зібраних даних в хмарному середовищі з використанням можливостей мов програмування C, C++, PHP та вільної системи керування реляційними базами даних MySQL.

Мета дослідження: розробка пристрою сигналізації для спостереження за станом спостережуваних об’єктів з передачею змін в хмарне середовище.

Пропонується метод вивчення конкурентних переваг дорожчих аналогів з проведенням аналізу їх компонентної бази та пошуком варіантів їх здешевлення та інтеграцією в більш дешевші рішення. Прикладна цінність отриманих результатів досягається завдяки реалізації розширеного функціоналу

корпоративних сигналізації на більш дешевшій компонентній базі. Подібний підхід дозволяє розширювати функціонал бюджетних рішень без значного збільшення їх вартості.

Запропонований підхід спрощення компонентної бази дозволяє використовувати більш примітивні набори радіо компонентів та логічних елементів зі збереженням функціоналу більш дорожчих рішень. Поставлені завдання передбачають вивчення конкурентних переваг аналогів, проектування пристрою та апаратних модулів, а також проектування та розробку програмного забезпечення для пристрою та серверної компоненти.

Розроблений пристрій відмінно виконує завдання сигналізації та багатоканального оповіщення при виникненні тривоги. Розроблене програмне забезпечення успішно виконує завдання синхронізації параметрів та обміну повідомленнями про тривогу.

Практичне значення отриманих результатів полягає в можливості використовування розробленої системи для створення тестів та тестування користувачів в різних установах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Розгляд стану проблеми

Головне призначення охоронної сигналізації полягає в оперативному і гарантованому сповіщенні власників або правоохоронні служби про несанкціоноване проникнення в приміщення, що охороняються. Рішення даної задачі можливе тільки при оснащенні об'єкту охорони сучасними високонадійними технічними засобами охоронної сигналізації.

Охоронна сигналізація (ОС) — це електронний пристрій, який дозволить завжди бути упевненим в безпеці будинку, офісу, квартири, складу, виробничого приміщення і т. д. Охоронна сигналізація розрахована на попередження несанкціонованого доступу в приміщення.

Системи охоронної сигналізації можна умовно розділити на два типи, взявши за основу такий критерій, як спосіб сповіщення про тривогу: [1].

Охоронна сигналізація автономна – основним призначенням автономної системи охорони є зупинка правопорушників шляхом активізації строб спалахів, сирен та ін. При спрацьовуванні даного виду охоронної сигналізації не відбувається оповіщення відповідних органів або власників, тобто передача сигналу тривоги не передбачена. Установка такої охоронної сигналізації підходить для організацій зі штатом співробітників, що перебувають на об’єкті постійно. Але найбільший ефект від охоронної сигналізації досягається при підключенні її на пульт позавідомчої охорони або на пульт централізованого спостереження приватного охоронного підприємства. Системи охоронної сигналізації з підключенням до пульту спостереження охоронного підприємства призначені для захисту об'єктів за допомогою установки в них спеціальних датчиків і виконуючих пристроїв, роль яких можуть виконувати комунікаційні або GSM модулі. В даному випадку, при спрацьовуванні датчика, охоронна система передає сигнал тривоги на центральний пульт нагляду. Згідно з вимогами, висунутими до охорони кожного окремо взятого об'єкта, можуть застосовуватися

різні види датчиків, які призначені для збору інформації. Датчики мають різні способи передачі інформації на ЦПУ, принципи дії чутливих елементів і типи контрольованих фізичних параметрів.

Система охоронної сигналізації спрацьовує, коли активний сигнал про проникнення на територію, що охороняється, надходить на стаціонарний пост охорони або спеціалізовану охоронну структуру, обладнану пультом, що збирає сигнали з декількох об’єктів або хоч би власникові. Такі системи зазвичай називають «пультова система охоронної сигналізації». По всьому об’єкту, що охороняється, розташовані датчики, які передають сигнал тривоги на контрольну панель, а звідти сигнал поступає на охоронний пульт. Охоронна структура в найкоротші терміни зобов’язана прибути на територію, що охороняється і присікти протиправні дії зловмисника [2].

Установка сигналізації нині є необхідністю для більшості власників нерухомості. З метою заощадження, в першу чергу розглядають варіанти GSM сигнальних систем, придбаних у Інтернет магазинах. При використанні для передачі повідомлень тривоги GSM систем можлива передача таких повідомлень на стільникові телефони власників об'єкту, що охороняється. Системи охоронної сигналізації своєчасно оповіщають про несанкціоноване проникнення на територію, що охороняється, і повідомлять про аварійні ситуації. Крім того, при проникненні всередину житла або офісу система може повідомити про тривогу телефонним дзвінком на вказаний заздалегідь номер. Для знімання інформації слугують різні сенсори: інфрачервоні і радіохвильові датчики руху, магнітні датчики відкриття дверей і вікон, акустичні датчики розбиття скла, датчики удару. Базовим блоком є зазвичай контрольні панелі, на які зводиться вся інформація від датчиків. Якщо є домашня тварина, датчики руху можна налаштувати так, щоб вони не реагували на переміщення тварини, але завжди надійно спрацьовували на появу людини. Промислові сигналізації можуть за допомогою спеціальних датчиків може відстежити протікання води і витік газу, і за заздалегідь налаштованою програмою віддати команду виконавчим пристроям комплексної

системи управління і охорони на виконання тієї або іншої дії (перекрити кран, включити сирену і тощо). Системи периметрової сигналізації беруть під охорону не тільки приміщення, але і прилеглу територію по периметру. Існують також гібридні варіанти, що включають в себе також функціонал пожежної сигналізації.

Пожежна сигналізація (ПС) — сукупність технічних засобів, призначених для виявлення пожежі, обробки, передачі в заданому вигляді повідомлення про пожежу, спеціальної інформації та видачі команд на включення автоматичних установок пожежогасіння і включення виконавчих установок систем протидимного захисту, технологічного та інженерного обладнання, а також інших пристроїв протипожежного захисту.

Установки і системи пожежної сигналізації, оповіщення та управління евакуацією людей при пожежі повинні забезпечувати автоматичне виявлення пожежі за час, необхідний для включення систем оповіщення про пожежу з метою організації безпечної (з урахуванням допустимого пожежного ризику) евакуації людей в умовах конкретного об'єкта.

Локальне оповіщення — проводиться за допомогою сирен, світлових, іноді мовних сповіщувачів.

Дистанційне мовне або текстове оповіщення — проводиться автоматичним надсиланням повідомлення каналами телефонного зв'язку (провідний або мобільний), через інтернет або локальні мережі тощо;

Зв'язок з апаратурою пульта централізованого спостереження (ПЦС) — за допомогою цифрових протоколів по дротових, оптоволоконних або бездротових каналах; раніше, в застарілих системах, використовувався дротяний зв'язок сигналами опору.

Сигнал зі встановлених у приміщеннях датчиків може надсилатись до кінцевого користувача, а також на приймально контрольний прилад (ППК), що знаходиться зазвичай поруч. Після отримання сигналу, пристрій надсилає його на пульт централізованого спостереження або власнику [3].

Сучасні домашні сигналізації у більшості випадків позбавлені можливостей підключення через кабельну мережу сумісних із стандартами ДСТУ та ГОСТ датчиків, багатоканального оповіщення, а також досить часто подібні розробки передають мережевий трафік через мережу Інтернет без належного шифрування. В ході роботи над данною роботою створено прототип, для розширення даних можливостей із застосуванням в бюджетних сигналізації без використання дорожчої компонентної бази.

1.2 Розгляд існуючих аналогів

Серед існуючих аналогів було проведено умовний поділ на такі групи: бюджетні сигналізації для дому, сигналізації для охорони підприємств, пожежні та промислові сигналізації. Пожежні сигналізації було виділено в окрему групу, оскільки вони безпосередньо не пов’язані з темою дослідження. Для подальшого дослідження було виділено наступні продукти.

ППКП Тірас — повноцінна сигналізація для корпоративного сектору, що має сертифіковану підтримку датчиків, виготовлених по стандартам ДСТУ. Дана сигналізація має також підтримку сертифікованих протоколів передачі повідомлень на центральний пульт охорони. Програмування та керування приладом здійснюється за допомогою чотирьох кнопок управління і вбудованої клавіатури. Пристрій може мати живлення від батареї. Додатково можливе підключення керованого модуля релейних виходів, а також модуля цифрового автодозвону. Крім того існує можливість підключити на один шлейф кілька пожежних сповіщувачів з контактами, на яких відбувається замикання/розмикання за допомогою реле, а також двожильні сповіщувачі, де одна і та сама пара проводів застосовується для живлення та оповіщення. Даний варіант є прикладом повноцінних систем для охорони підприємств. Однак для її встановлення необхідно наймати спеціально навчений персонал. Вартість покупки та обладнання такої сигналізації може виходити за межі 50 тисяч гривень, що значно ускладнює її отримання для даного дослідження. Для дослідження

можливостей системи було використано більш старший аналог, розроблений тією ж фірмою (ТОВ «Тірас 12») — ADT Orion 8K див на рис 1.1. Хоча на базі даного рішення відсутній GSM модуль, однак в ній натомість наявний додзвон по аналоговій телефонній лінії. [4].

Рисунок 1.1 — ADT Orion 8K

Бездротова сигналізація GSM Kerui G18 див на рис 1.2.

Повноцінна сигналізація яка всвою власність зсередини та отримуйте мобільні сповіщення при відкритті дверей або вікон та при виявленні руху в домашніх умовах.

При вторгненні сигналізація G18 видасть попередження сиреною 110 дБ, попередить сусідів та стримає потенційних зловмисників, а також здійснить голосовий дзвінок та смс на ваші 6 телефонних номерів. Детальніші характеристики:

— 4-х діапазонний модуль GSM 900/1800/1900 МГц;

— 433 МГц цифрова мережа бездротових датчиків на сучасному кодеку — Learning Code Ev1527;

— робоча напруга — 5 мА;

— струм споживання до 55мА;

— в режимі тривоги до 450мА;

— резервна батарея 7.2 В, до 4—6 годин автономної роботи.

При зникненні живлення сигналізація має резервну батарею для роботи 8 годин. Дистанційне керування, у будь-який час та з будь-якого місця.

Завдяки дозволу з телефону можна легко керувати постановкою, зняттям з охорони та домашнього режиму, а також за допомогою простого пульта дистанційного керування. Але також є і недоліки постійна робота встановленої SIM карти оператор повинен її обслуговувати, що має на увазі поповнення балансу. І також рівень сигналу не у всіх куточках нашої країни стабільний, але ж від цього безпосередньо залежить швидкість передачі екстреного виклику.

Налаштування KERUI для потреб. Сумісна з бездротовим PIR датчиком, дверним датчиком, детектором диму, детектором газу, кнопкою паніки і т. д. Вона широко використовується в будинку, фабриці, школі, магазині, віллі та житловому районі.

Сигналізація може дзвонити на інші телефони безпосередньо як звичайний телефон. Запланована постановка/зняття з охорони. Якщо важко ставити/знімати сигналізацію щодня. Є 4 набори запланованих функцій arm/disarm/stay arm по зоні/дню/часу. Установка на запит, автоматична авто постановка.

Керувати сигналізацією також можна віддалено, наприклад, дзвінком, смс або через програму на iOS/Android. Системою ведеться журнал подій, є можливість включення та відключення за розкладом. Ваш будинок ― Ваша фортеця, і дана сигналізація допоможе вам у його повноцінному захисті. Наявність "Тривожної кнопки" на пультах дистанційного керування дозволяє моментально активувати сирену. [5].

Рисунок 1.2 — Повна комплект бездротової сигналізація GSM Kerui G18

ATIS Kit 200T див на рис 1.3 бездротовий комплект автономної Wi-Fi сигналізації.

До складу комплекту входить: «розумна» сирена централь, бездротовий ІЧ-датчик руху, бездротовий датчик відчинення дверей, 1 радіо брелок для постановки та зняття з охорони, блок живлення, елементи кріплення, інструкція. 24 бездротові зони, 8 пультів керування. Робоча частота 433МГц, протокол зв'язку Ademco control ID. Макс. кількість пристроїв, що підключаються: 24. Керування через мобільний застосунок.

Завдяки широкому набору датчиків и додатковому обладнанню, що входять в комплект постачання (датчик руху, відчинення дверей, сирена), бездротова сигналізація ATIS Kit 200T «з коробки» готова для забезпечення ефективної охорони квартири, котеджу, дачі, гаражу чи офісу.

Особливості ATIS Kit 200T:

— вбудований Wi-Fi модуль для віддаленого керування охоронною системою;

— 24 бездротові зони можуть бути встановлені в режимах під охороною (Arm), Режим часткової охорони (Home Arm), охорона знята (Disarm);

— можливість підключення 8 пультів керування;

— 20 користувачів, які можуть керувати централлю через мобільний застосунок;

— сповіщення про відсутність Wi-Fi з'єднання;

— підтримка Amazon Alexa, Google Assistant;

— регулювання гучності сирени;

— наявність тамперу;

— вбудований літієвий акумулятор забезпечує безперебійну роботу пристрою при вимкненні живлення (за відсутності постійного живлення система сповістить звуковим сигналом).

Видимими недоліками даного рішення є відсутність можливості налаштувати сигналізацію без додатку для смартфону та хмари виробника, а

також відсутність GSM оповіщення. Пристрій містить вбудований блок живлення з мережі змінного струму. Ознайомлення з даним рішенням проводилось безпосередньо з використанням його апаратної та програмної реалізації. [6].

Рисунок 1.3 — Повна комплект сигналізація ATIS Kit 200T

На основі огляду було визначено конкурентні переваги кожного продукту: ППКП Тірас — сумісність з датчиками ДСТУ, сигналізація GSM Kerui G18 — використання зв’язку GSM для передачі повідомлень тривоги, ATIS Kit 200T — використання технології хмари для оповіщення та конфігурації пристрою. Технічні особливості даних переваг описано у розділі 2 даної роботи.

1.3 Вибір засобів дослідження та розробки

Для виконання завдань дослідження можливостей апаратних модулів, розробки та тестування електричних схем, а також для розробки програмного забезпечення пристрою та серверної компоненти.

Для дослідження можливостей та відлагодження апаратних радіо модулів було обрано цифровий приймач RTL SDR та програму Airspy SDR. Даний набір апаратного та програмного забезпечення дозволяє визначати потужність радіосигналу цифрових передавачів а також містить потужний набір декодерів.

Для відслідковування мережевого IP чи Ethernet трафіку, що передається контролером було обрано програму Wireshark.

Wireshark — це дуже потужний і один із найкращих у світі сніффер для захоплення та декодування мережевого трафіку. Надає можливість декодувати понад 500 різних протоколів мереж передачі даних та телекомунікаційних протоколів, включаючи протоколи стільникового зв'язку. Він є де факто (і часто де юре) стандартом у багатьох галузях промисловості та освітніх установах у всьому світі. Багато виробників комерційних продуктів використовують його у своїх рішеннях як декодувальник.

Програма WireShark абсолютно безкоштовна і постійно доопрацьовується кількома авторами на пожертвування від спонсорів із далекого 1998 року [7].

Proteus Design — пакет програм для автоматизованого проектування (САПР) електронних схем. Розробляється компанією Labcenter Electronics. Пакет являє собою систему схемотехнічного моделювання, що базується на основі моделей електронних компонентів, прийнятих в PSpice.

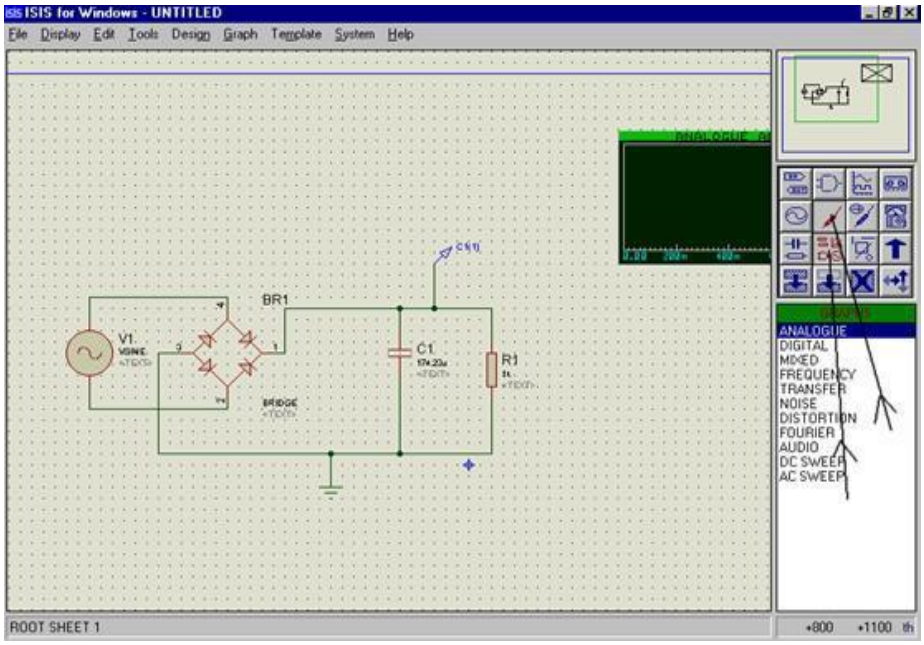


Рисунок 1.4 — Користувацький інтерфейс ISIS Proteus

Відмінною рисою пакету Proteus Design є можливість моделювання роботи програмованих пристроїв: мікроконтролерів, мікропроцесорних систем, DSP і ін. Proteus Design включає в себе більше 6000 електронних компонентів з усіма довідковими даними, а також демонстраційні ознайомчі проекти. Саме завдяки підтримці симуляції популярних мікроконтролерів та простоті використання було обрано даний пакет для моделювання схем.

Серед таких мов програмування як C/C++, AVR Assembler та Python написання коду розробки пристрою було обрано мову програмування C/C++. Такий вибір було зроблено через використання платформи Arduino, яка в свою чергу базується на мові програмування C/C++ та компіляторі AVR GCC. Даний компілятор Arduino IDE значно спрощує написання програм для цієї платформи і створення пристроїв на Arduino стає набагато легшим, навіть при відсутності значних знань мови C/C++[8].

Arduino IDE — середовище розробки, що дозволяє писати свої програми (так звані скетчі) для платформи Arduino. Ця платформа в першу чергу орієнтується на конструкторів аматорів, які застосовують Arduino для побудови простих систем автоматики і робототехніки. Інтегроване середовище розробки для Windows, MacOS та Linux, розроблене на Сі та C++, призначене для створення та завантаження програм на Arduino сумісні плати, а також на плати інших виробників.

Вихідний код для середовища випущено під загальнодоступною ліцензією версії GNU 2. Підтримує мови Сі та C++ з використанням спеціальних правил структурування коду. Arduino IDE надає бібліотеку програмного забезпечення з проекту Wiring[en], яка надає безліч загальних процедур введення та виведення. Для написаного користувачем коду потрібні лише дві базові функції для запуску ескізу та основного циклу програми, які скомпільовані та пов'язані з заглушкою програми main у циклічну програму з ланцюжком інструментів GNU, також включеною в дистрибутив IDE.

Використовує програму Arduino для перетворення коду, що виконується, в текстовий файл у шістнадцятковому кодуванні, який завантажується в плату Arduino програмою завантажувачем у вбудованому програмному забезпеченні. Зі зростанням популярності Arduino інші постачальники в якості програмної платформи почали впроваджувати компілятори та інструменти з відкритим вихідним кодом (ядра), які можуть створювати і завантажувати ескізи в інші мікроконтролери, що не підтримуються офіційною лінійкою мікроконтролерів Arduino.

У жовтні 2019 року організація Arduino почала надавати доступ до нової Arduino Pro IDE з налагодженням та іншими розширеними функціями [8].

Для написання серверної компоненти, серед таких мов програмування як C# (ASP.NET), PHP, Java, JavaScript, Python та Ruby було обрано PHP через ряд переваг:

— можливість створення HTML документів із вбудованими командами PHP;

— php скрипти виконуються виключно на стороні сервера;

— можливість створювати якісні Web додатки за дуже короткі терміни;

— php є наявною потужна підтримка баз даних MySQL, що дозволяє швидко адаптовувати бази даних для цілей розробки серверної компоненти;

— php є простим для освоєння, що робить його ідеальним кандидатом для застування в даній роботі.

На користь вибору мов програмування C/C++ та PHP вплинув суб’єктивний фактор володіння даними мовами автором даної роботи.

1.4 Визначення завдань дослідження

Робота над проектом передбачає проектування та розробку макетного пристрою, який за рахунок використання недорогої компонентної бази зможе набути переваг більш дорогих аналогів.

Процес роботи над проектом передбачає виконання таких завдань:

розгляд стану проблеми та існуючих аналогів;

— визначення завдань дослідження;

— вивчення конкурентних переваг аналогів;

— проектування пристрою та апаратних модулів;

— проектування та розробка програмного забезпечення для пристрою та серверної компоненти.

2 ПРОЕКТУВАННЯ БЛОКІВ СИСТЕМИ БЕЗПЕКИ

2.1 Дослідження конкурентних апаратних переваг модулів зв’язку

В даному розділі розглянуто процес проектування окремих модулів пристрою. В розділі приділяється увага побудові архітектури всього контролера сигналізації, а також методи його поєднання із комп’ютерними мережами.

В ході виконання дослідження аналогів особливу увагу було приділено тому, як аналоги здійснюють зв'язок з терміналами користувача, які зможуть передати сигнал тривоги та конкретизувати причину його виникнення. Найбільш ефективними нині універсальними терміналами приймання сповіщень є мобільні пристрої.

До мобільних пристроїв можна віднести:

— смартфони;

— планшетні ПК;

— смарт браслети з вібрацією та дисплеєм.

Існує досить велика кількість технологій та протоколів по яким за допомогою бездротового зв’язку можна передати сигнал власникові об’єкту:

— bluetooth;

— sms сповіщення;

— push сповіщення;

— електронна пошта;

— боти для месенджерів;

— спеціальні мобільні додатки.

Під час визначення придатності технологій вимогам стабільності та гарантування зв’язку зі списку було виключено Bluetooth, через невелику дальність передачі повідомлень, а електронна пошта є занадто складною технологією для реалізації та налаштування. Крім того концепція електронної пошти не передбачає доставки повідомлень «прямо в руки» (тобто зазвичай потрібно оновлювати список листів в браузері, щоб дізнатись чи є нові листи).

Натомість використання SMS та повідомлень месенджерів є більш виправданим, через те, що вони підсвічуються смартфонами, а спеціальні Android додатки самостійно можуть встановлювати спосіб сповіщення власника.

З концептуальної точки зору, технологія передачі SMS повідомлень має суттєві відмінності від таких технологій, як передача сповіщень через боти месенджерів та повідомлення мобільних додатків. У випадку передачі SMS використовується GSM інфраструктура мобільних операторів. Мережа Інтернет в даному випадку якщо і використовується, то лише в глибинах операторів. Перевагою такого методу передачі даних є використання ланцюгового принципу передачі повідомлень.

Приклад схема ланцюгової передачі повідомлень через мережу GSM приведено на рисунку 2.1

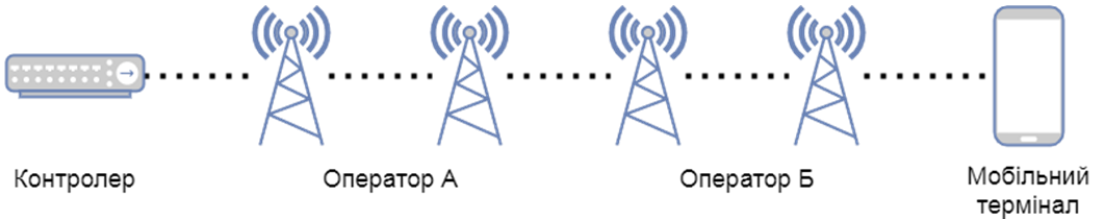


Рисунок 2.1 — Схема ланцюгової передачі повідомлень через мережу GSM

Даний вид сигналізації дозволяє здійснювати дистанційне спостереження за гаражем, знаходячись на будь якій відстані. Усю інформацію про стан об'єкта можна прослуховувати або зорово спостерігати через стільниковий телефон. Також існує можливість отримання SMS повідомлень.

Коротка схема роботи системи виглядає так. Дані про проникнення в приміщення надходять від датчиків центральний блок (контролер). Опрацьований контролером сигнал надходить у телефон власнику нерухомості або охоронній фірмі у вигляді дзвінка або SMS повідомлення.

Приклад принцип роботи GSM мережі приведено на рисунку 2.2

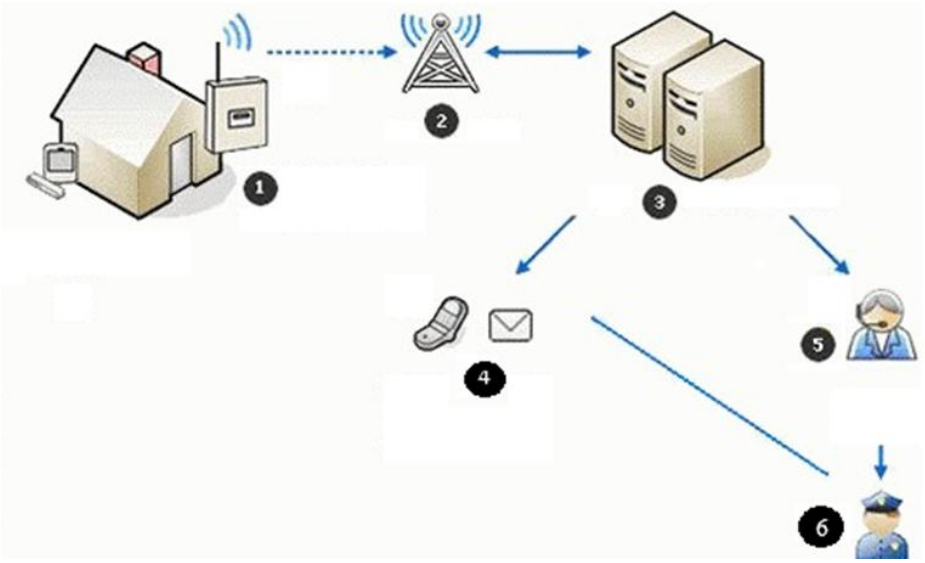


Рисунок 2.2 — Принцип роботи GSM мережі

GSM мережа складається:

— безпровідні датчики і контрольна панель;

— GSM сигнал;

— центр обробки GSM;

— дзвінок або SMS в реальному часі;

— дзвінок в CALL CENTER;

— виклик поліції або аварійної служби.

Такі два варіанти, такі як передача сповіщень через боти месенджерів та повідомлення мобільних додатків передбачають асинхронний тип передачі сигналу, який спочатку приходить на сервер у хмарному середовищі, а лише потім «підхоплюється» та доставляється на пристрій відповідним додатком (рис. 2.3).

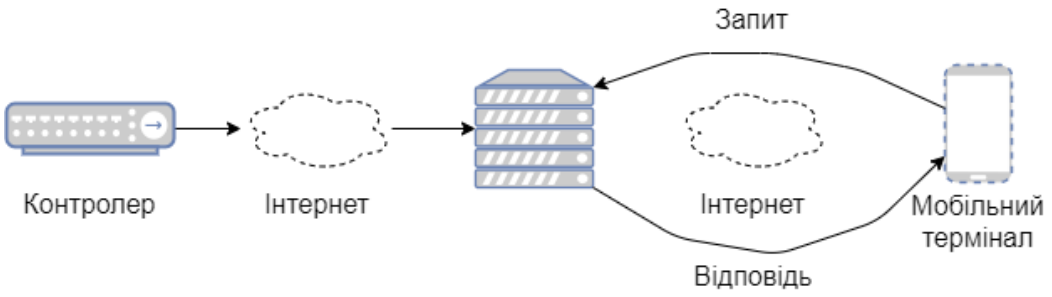


Рисунок 2.3 — Схема передачі повідомлень через мережу Інтернет

Принцип роботи бездротової мережі побудований на використанні радіохвиль, а сам обмін даними багато в чому нагадує переговори з використанням радіозв'язку. Адаптер бездротового зв'язку формує інформацію в радіосигнал і передає його в ефір через антену. Бездротовий маршрутизатор приймає і робить зворотне перетворення сигналу. Далі інформація направляється маршрутизатором в мережу Інтернет по кабелю від провайдера.

2.2 Проектування бази даних

У результаті проектування має бути визначена структура бази, тобто склад таблиць, їхня структура та логічні зв'язки. Структура реляційної таблиці визначається складом стовпців, їхньою послідовністю, типом даних кожного стовпця та їхнім розміром, а також ключем таблиці. Процес проектування можна здійснювати двома підходами. За першого підходу спочатку визначають основні задачі, для розв'язання яких створюється база, та потреби цих задач у даних. За другого підходу визначають предметну область, здійснюють аналіз її даних і встановлюють типові об'єкти предметної області. Найбільш раціональним підходом проектування бази даних є поєднання обох підходів. База даних сукупність взаємозв'язаних даних, які знаходяться під управлінням СУБД .

Модель даних – це деяка абстракція, яка, будучи застосовна до конкретних даних, дозволяє користувачам і розроблювачам трактувати їх уже як інформацію, тобто відомості, що містять не тільки дані, але й взаємозв’язок між ними [16].

Рисунок 2.4 — Класифікація моделей даних за архітектурою бази даних

Система управління базами даних (СУБД) — сукупність мовних і програмних засобів, які призначені для створення, ведення і сумісного використання БД багатьма користувачами.

Переваги застосування СУБД:

— мінімізація збитковості даних;

— несуперечливість даних і контроль їх цілісності;

— несуперечливість даних і контроль їх цілісності;

— незалежність прикладних програм від даних;

— підвищена безпека;

— розвинені засоби резервного копіювання і відновлення;

— багатокористувацький режим роботи.

Недоліки застосування СУБД:

— використання значної частини ресурсів на потреби СУБД, а не на прикладну задачу;

— вартість СУБД;

— підвищені вимоги до технічного і програмного забезпечення продуктивність;

— підвищені вимоги до кваліфікації робітників;

— наслідки збоїв.

Для забезпечення функціонування наступних типів сутностей буде проектуватись структура бази даних:

— користувачі;

— пристрої користувачів;

— об’єкти пристроїв;

— події, що відбуваються з об’єктами;

— вирішення цієї проблеми на стадії проектування полягає у такому;

— наявність обов'язкових і необов'язкових значень даних для атрибутів (NULL, NOT NULL);

— наявність обмежень для доменів атрибутів (визначення області значень або діапазону значень);

— цілісність сутностей (обов'язкова наявність Primary Key в кожному відношенні);

— посилкова цілісність (зв'язування таблиць за допомогою Foreign Key);

— обмеження предметної області (бізнес правила), які реалізуються як засобами БД, так і на рівні застосувань.

У табл. 2.1 наведені правила зовнішнього ключа для відношення "один до багатьох" для сильної сутності.

Таблиця 2.1 — Підтримка посилкової цілісності для сильної сутності

Тип зв'язку	Вимоги до зовнішнього ключа
Обов'язкова наявність значень відповідних екземплярів у батьківській і залежній таблицях.	NOT NULL ON DELETE RESTRICT ON UPDATE CASCADE
Необов'язкова наявність значень відповідних екземплярів у батьківській і залежній таблицях.	NULL ALLOWED ON DELETE SET NULL ON UPDATE CASCADE
Обов'язкова наявність значень відповідних екземплярів у залежній таблиці і необов'язкова наявність значень в батьківській таблиці.	NULL ALLOWED ON DELETE SET NULL ON DELETE RESTRICT ON UPDATE CASCADE
Обов'язкова наявність значень відповідних екземплярів у батьківській таблиці і необов'язкова наявність значень в залежній таблиці.	NOT NULL ON DELETE RESTRICT ON UPDATE CASCADE

Одним з найбільш важливих напрямів розвитку баз даних є зближення технологій баз даних, WEB технологій. Середовище WEB забезпечує доступ до великих обсягів інформації в Інтернеті, однак крім накопичення слабко структурованої інформації необхідно забезпечувати ефективне управління цією різноманітною інформацією. Розробка платформи XML забезпечує нові можливості для ефективного доступу до інформаційних ресурсів цього середовища, а також для створення нових технологій інтеграції ресурсів.

Поєднання технологій баз даних і WEB технологій дозволяє вирішувати такі задачі:

— організація взаємозв'язку СУБД, які працюють на різних платформах;

— використання в Інтернеті інформації з існуючих локальних мережних баз даних (публікація баз даних);

— застосування СУБД для впорядкування і каталогізації інформації в Інтернеті;

— застосування в інформаційних систем в Інтернеті на основі багаторівневої архітектури баз даних;

— застосування мови *SQL* для пошуку інформації;

— використання засобів СУБД для забезпечення безпеки;

— даних, управління транзакціями при доступі до інформації в Інтернеті;

— стандартизація користувацького інтерфейсу на основі застосування оглядачів. WEB, використання оглядача WEB в якості дешевої клієнтської програми доступу до бази даних.

Наступним напрямом в розвитку СУБД є розвиток і подальше вдосконалення управлінням об'єктно орієнтованими базами даних. Сучасні виробники СУБД або будують свої включають в свої програмні продукти об'єктно реляційні можливості [17].

Для ER моделі існує алгоритм однозначного перетворення її в реляційну модель даних. ER модель — це представлення бази даних у вигляді наочних

графічних діаграм. ER модель візуалізації процес, що визначає деяку предметну область.

Діаграма “сутність зв’язок” — це діаграма, яка представляє в графічному вигляді сутності, атрибути і зв’язки.

Як видно з типів сутностей, кожна з них має залежність від попередньої. На базі цих залежності було створено структуру типів записів, яку подано у вигляді ER діаграми рис. 2.5. [18].

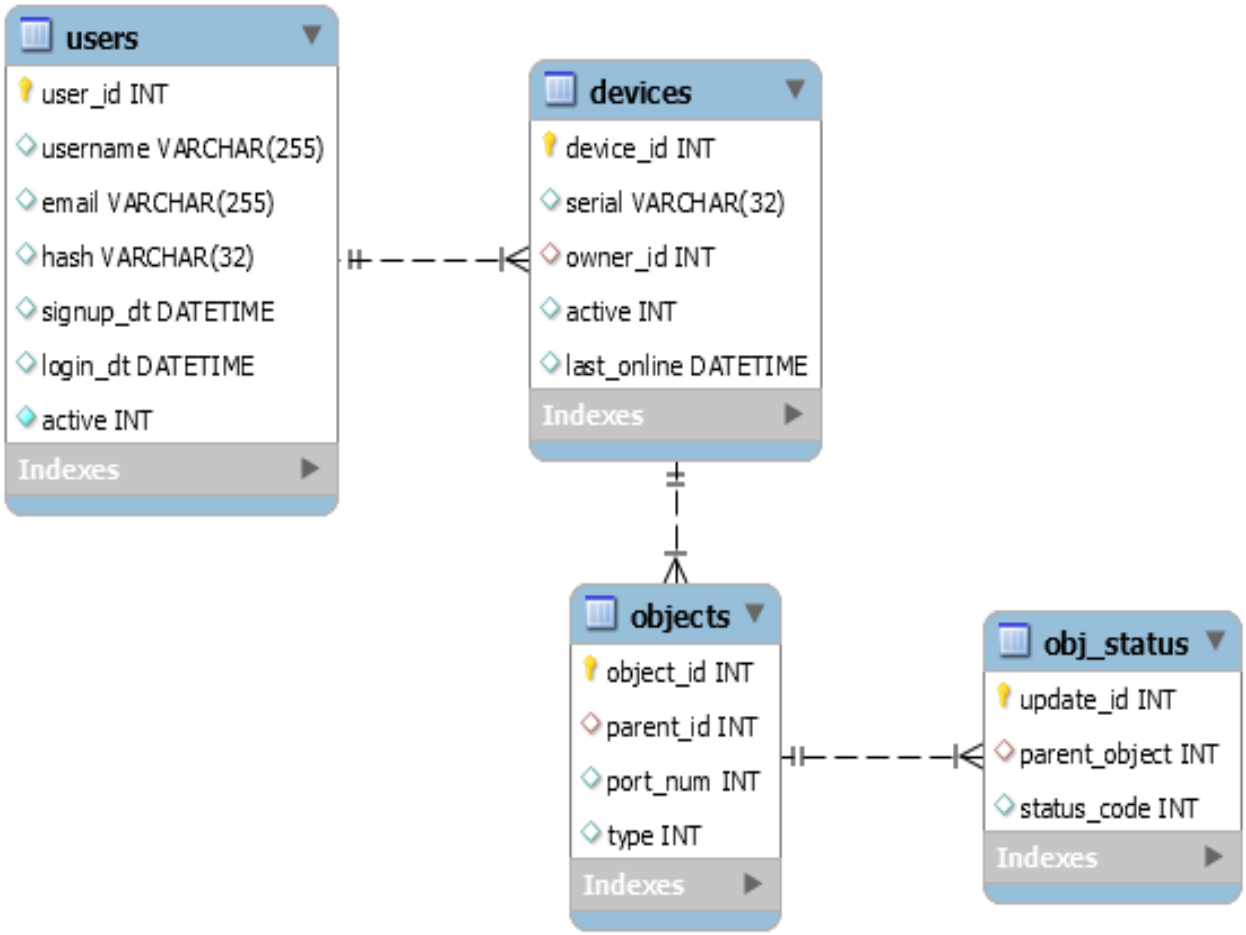


Рисунок 2.5 — ER діаграма бази даних для серверної компоненти

2.3 Проектування архітектури системи безпеки

Для забезпечення багатоканальної передачі повідомлень тривоги з пристрою було розглянуто варіанти з використання терміналу прийому сигналів

тривоги у вигляді смартфону. З врахуванням конкурентних можливостей по прийому сигналу з датчиків було визначено такі основні інтерфейси зчитування станів датчиків:

— аналогові провідні датчики;

— бездротові датчики (433 МГц);

— датчики мікроклімату.

Для обміну отриманими даними з іншими термінальними пристроями було визначено такі можливі термінали:

— мобільні пристрої;

— брілки керування сигналізацією (433 МГц);

— веб інтерфейс.

В цілях реалізації підтримки вищевказаних типів терміналів було визначено наступні апаратні інтерфейси:

— модуль зв’язку з мережею WiFi або Ethernet;

— GSM модуль;

— радіо модуль для отримання та передачі інформації на брілки (433 МГц).

Мікроконтролер (Micro Controller Unit, MCU) — мікросхема, призначена для управління електронними пристроями. Типовий мікроконтролер поєднує на одному кристалі функції процесора і периферійних пристроїв, містить ОЗП і ПЗП. По суті, це однокристальний комп'ютер, здатний виконувати досить прості завдання. Відрізняється від мікропроцесора інтегрованими в мікросхему пристроями введення виведення, таймерами і іншими периферійними пристроями [10].

За основу для розробки структурної схеми було взято базовий принцип роботи інтерфейсів мікроконтролера тільки вхід, тільки вихід на обидва рис. 2.6 [11].

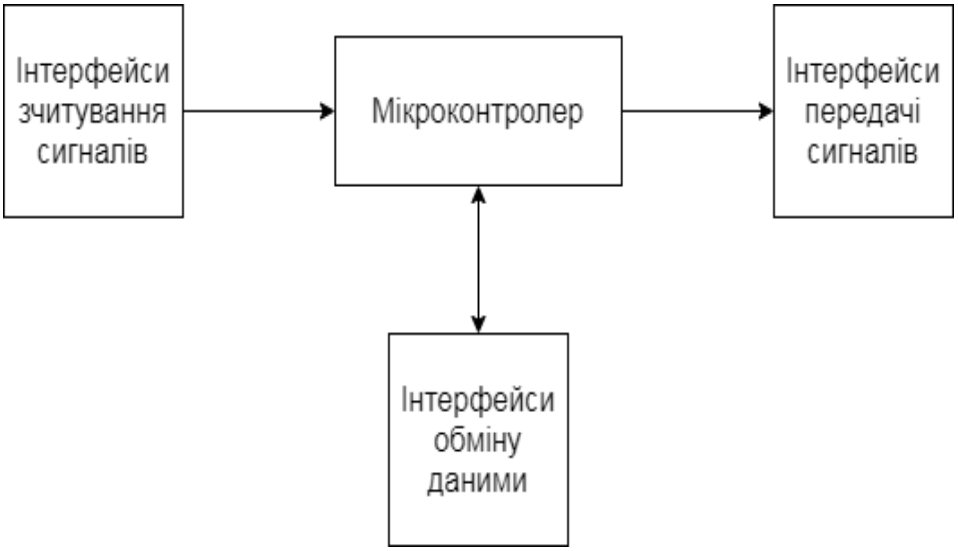


Рисунок 2.6 — Принцип підключення сторонніх модулів до інтерфейсів контролера.

В ході опрацювання та відбору сумісних модулів для Arduino Nano та переліку необхідних інтерфейсів було створено результуючу структурну схему рис. 2.6.

Arduino Nano — це повнофункціональний мініатюрний пристрій на базі мікроконтролера ATmega328 (Arduino Nano 3.0) або ATmega168 (Arduino Nano 2.x), адаптований для використання з макетної плати.

За функціональністю пристрій схожий на Arduino Duemilanove, і відрізняється від нього розмірами, відсутністю роз'єму живлення, а також іншим типом (Mini-B) USB кабелю. Arduino Nano розроблено і випускається фірмою Gravitech.

Етапи роботи мікроконтролера Atmel ATmega168 або ATmega328 наведено в Таблиці 2.2

Arduino Nano може живитися через кабель Mini B USB, від зовнішнього джерела живлення з нестабілізованою напругою від 6В до 20В (через вивід 30В) або зі стабілізованою напругою 5В (через вивід 27В). Пристрій автоматично вибирає джерело живлення з більшою напругою.

Таблиця 2.2 — Характеристики Arduino Nano

Мікроконтролер	Atmel ATmega168 або ATmega328
Робоча напруга (логічний рівень)	5В
Напруга живлення	від 7В до12В
Напруга живлення гранична	від 6В до 20В
Цифрові входи\виходи	14 (з яких 6 можуть використовуватися як ШІМ виходи)
Аналогові входи	8
Максимальний струм одного виводу	40 мА
Flash пам'ять	16 КБ (ATmega168) або 32 КБ (ATmega328) з яких 2 КБ використовуються завантажувачем
SRAM	1 КБ (ATmega168) або 2 КБ (ATmega328)
EEPROM	512 байт (ATmega168) або 1 КБ (ATmega328)
Тактова частота	16 МГц
Розміри плати	1.85 см x 4.3 см

Напруга на мікросхему FTDI FT232RL подається тільки в разі живлення Arduino Nano через USB. Тому при живленні пристрою від інших зовнішніх джерел (НЕ USB), вихід 3.3В (формований мікросхемою FTDI) буде неактивний, в результаті чого світлодіоди RX і TX можуть мерехтіти при наявності високого рівня сигналу на виводах 0 і 1[12].

Побудована структурна схема зображена у Додатку В вона здатна забезпечити такі базові функції як: циклічне отримання стану датчиків, отримання

команд з брілка (постановка та зняття з захисту). Блок оповіщення дозволяє реалізовувати одразу три способи передачі повідомлень: на брілок керування, через SMS, через Інтернет (за допомогою WiFi або Ethernet модуля).

Варто відзначити, що зображений мережевий Ethernet модуль може також бути продубльований з WiFi модулем, якщо це передбачено бюджетом комплектації.

2.4 Розробка модулю зчитування станів провідних шлейфів

Для роботи з провідними шлейфами варто завжди мати на увазі, що кожен шлейф знаходить у справному стані, і є можливим отримання з нього правильних сигналів.

В реальності як самі датчики так і їх шлейфи можуть зазнавати фізичних ушкоджень, і як наслідок, втрачати справність.

Пристрій повинен бути у змозі самостійно проводити самодіагностику всіх шлейфів. В ході реалізації програмного забезпечення (розділ 4) було визначено чотири стани кожного провідного шлейфу:

— розрив лінії;

— датчик відкритий (тривога);

— датчик закритий;

— коротке замикання.

На основі даних станів було створено таблицю, де цифрові входи A і B будуть зчитуватись контролером Arduino.

Етапи роботи зчитуватись станів датчиків контролером Arduino наведено в Таблиці 2.3.

У деяких випадках пошкодження шлейфів або датчиків може призвести до короткого замикання. Оскільки найбільш простим способом визначення цих станів є вимір вхідної напруги, тому завдяки ньому можна визначати початковий опір шлейфу (формула 2.1).

Таблиця 2.3 — Зчитування станів датчиків

Вхід/Стан	Розрив	Відкритий (Тривога)	Закритий	К/З
A	0	0	1	1
B	0	1	0	1

Відомо, що

$$U = I_{ш}(R_1 + R_2 + R_3) \quad (2.1)$$

де $I_{ш}$ — загальний струм шлейфу (визначається окремо для кожного датчика);

R_1 — вхідний опір контролера (визначається відповідно до потреб захисту від короткого замикання);

R_2 — опір датчика у закритому стані;

R_3 — виникає, коли датчик відкривається, коли датчик закритий, вважається, що $R_3 = 0$.

Датчики безпеки — першочерговий засіб забезпечення безпеки на ділянці, де людина і машина працюють разом. Володіючи інтелектом, ці датчики припиняють роботу машини в ситуаціях, небезпечних для людини. Основна перевага сучасних датчиків полягає в тому, що сигнал поступає не лише на пульт управління служби безпеки, але й на мобільний пристрій відповідальної особи (власника приміщення) відразу ж після виявлення небезпечної події.

Пристрої для системи безпеки бувають найрізноманітніших видів: датчики руху, відкриття дверей, розбиття скла, виявлення газу, диму, води, вібрацій й інші.

Датчик вібрації — високочутливий прилад, який реагує на будь-які, навіть незначні вібрації поверхні. Активно застосовується для охорони будівель.

Датчик затоплення — сповіщає про витік води з водопроводу, опалювальної системи, каналізації. Пристрій підключається через кабель або мережу і швидко реагує на витік води. Деякі системи можуть самостійно перекривати воду.

Датчик газу — сигналізує про витік газу, реагує на підвищення рівня чадного газу в приміщенні. При підвищенні концентрації газу в повітрі пристрій видасть звукове попередження й відправить повідомлення на телефон або комп'ютер власника.

Пожежні датчики, датчики температури і задимлення — визначають рівень концентрації диму в повітрі або підвищення температури. В той час, як датчик диму реагує на рівень задимлення, температурні оповіщають про перевищення температурного порогу або швидкості підвищення температури в приміщенні. Такі пристрої дозволяють ще на ранній стадії усунути займання і евакуювати людей. Це надійний спосіб уникнути пожежі.

Датчик руху — універсальний пристрій, який визначає переміщення об'єктів по території. У ньому встановлена оптична система, яка дозволяє виявляти переміщення людей в радіусі дії датчика. Такі прилади встановлюють в декількох місцях будинку, офісу або підприємства.

Шлейф сигналізації і топології на рис 2.8 — замкнутий петлевий шлейф, що забезпечує максимальну надійність системи.

У разі обриву або одиночного КЗ кільцевої шлейф перетвориться в два розімкнутих, забезпечуючи інформаційний доступ до компонентів системи з двох сторін.

Всі сповіщувачі шлейфу мають адреси, які встановлюються автоматично в режимі «Адресація» або вручну за допомогою пульта ручного адресації (ПРА).

Прилад розрізняє сповіщувачі і модулі введення — виведення і надає діапазон адрес від 001 до 127 для сповіщувачів і діапазон від 129 до 229 для модулів вводу — виводу. Адреси сповіщувачів і пристроїв введення — виведення розподіляються послідовно від початку і до кінця шлейфа [13].

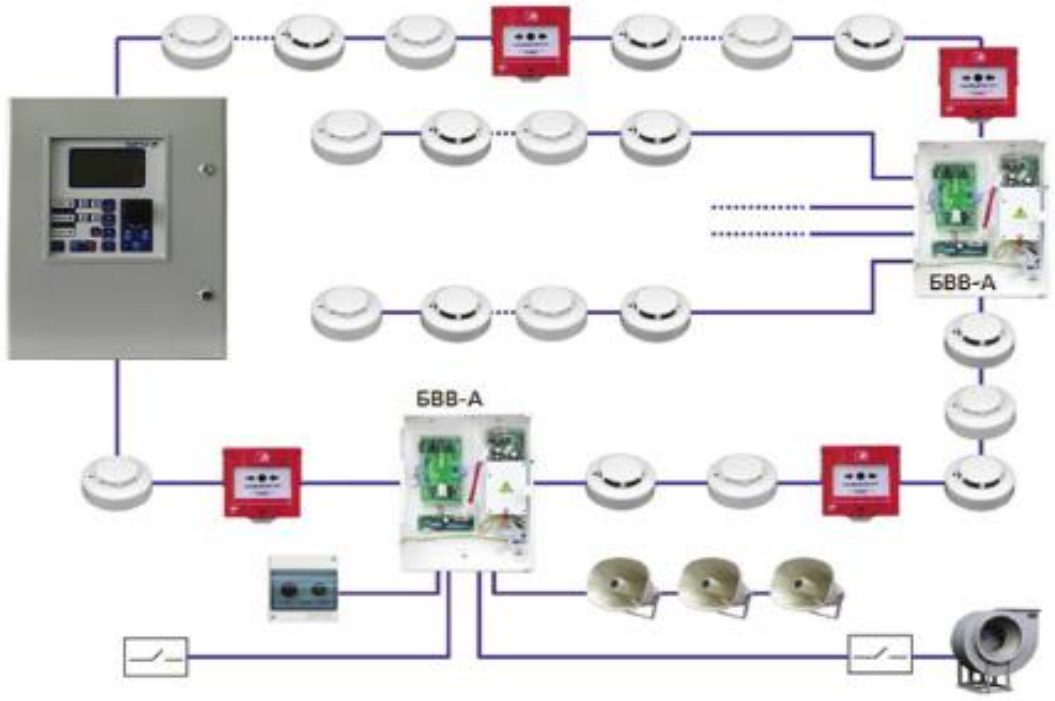


Рисунок 2.7 — Приклад топології шлейфу

Для кожного шлейфу в ході опитування вводів буде визначатись опір, та відповідно до графіку буде визначатись стан (рис. 2.7).

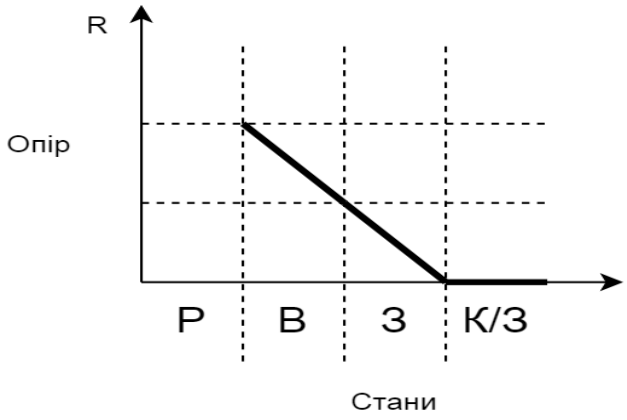


Рисунок 2.8 — Графік залежності станів від опору шлейфу

Схема паралельного та послідовного підключення датчиків у шлейфу зображено (на рис 2.9) до клеми шини один у нас підключення паралельні кожин датчик складається із R супутнього резистора і безпосередньо нормального

розімкнутого датчика нормальні розімкнуті датчики це датчик Д1 Д2 Д3 в кінці шина зєднана резистором $R_{Терм}$. Шина 1 яка повністю складається з нормального розімкненого датчиків. Шина 2 складається з нормально замкнених датчиків можуть бути наприклад герконові датчики які нормально замкнені Д4 Д5 Д6 кожен датчик паралельно резистор і в кінці шини є термінуючий резистор.

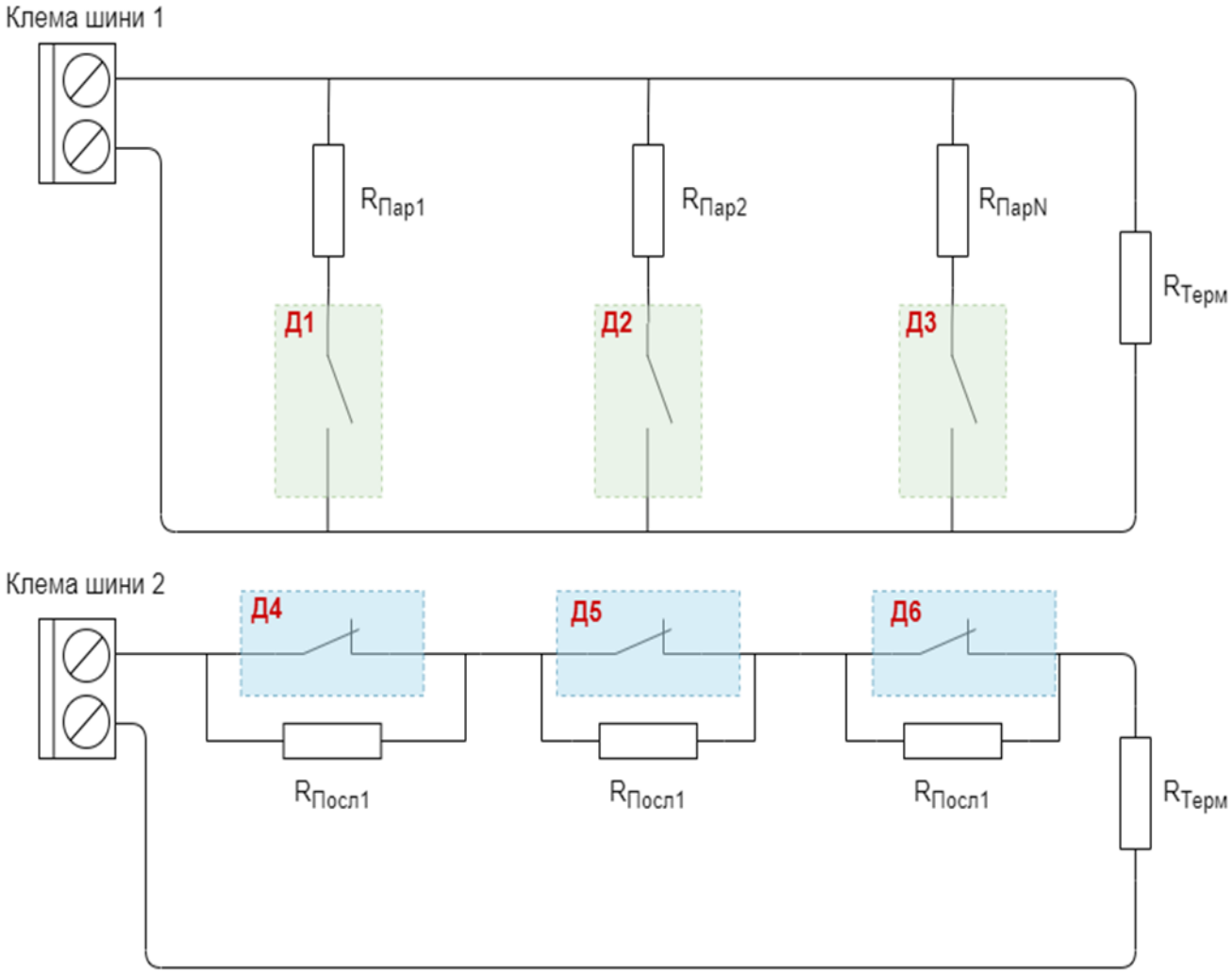


Рисунок 2.9 — Схема паралельного та послідовного підключення датчиків у шлейфу

Найбільш дешевими компонентами, на базі яких можна реалізувати замір вихідної напруги, є операційні підсилювачі. Одним з найбільш доступних є мікросхема LM741 (рис. 2.9), а також її чотирьох канальний аналог LM248, які можна використовувати для порівняння напруги та переводити результат в

цифрове значення (0 або 1). Даної мікросхеми достатньо для формування сигналу входу А.

Операційний підсилювач — це електронний підсилювач напруги з високим коефіцієнтом підсилення, що має диференціальний вхід та зазвичай один вихід. Напруга на виході може перевищувати різницю напруги на входах у сотні або навіть тисячі разів. Свій початок операційні підсилювачі ведуть від аналогових комп'ютерів, де вони застосовувалися у багатьох лінійних, нелінійних та частото — залежних схемах. Параметри схем з операційними підсилювачами визначаються тільки зовнішніми компонентами, а також невеликою температурною залежністю або розкидом параметрів при їх виробництві, що робить операційні підсилювачі дуже популярними елементами при конструюванні електронних схем.

Операційні підсилювачі є найбільш затребуваними приладами серед сучасних електронних компонентів, вони знаходять своє застосування в споживчій електроніці, застосовуються в індустрії та наукових приладах [14].

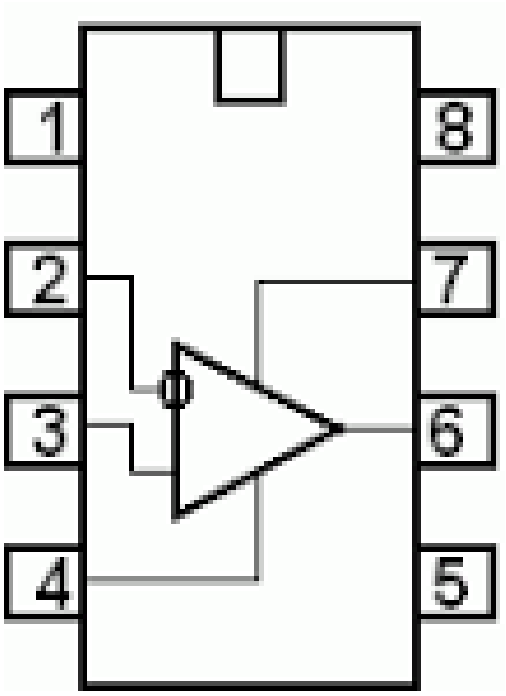


Рисунок 2.10 — Виводи мікросхеми LM741

Виводи мікросхеми LM741 виводяться:

1) установка нуля;
2) інвертуючий вхід;
3) неінвертуючий вхід;
4) живлення;
5) установка нуля;
6) вихід;
7) живлення +;
8) не приєднується.

Для формування значення входів шин (контроль технічної тривоги), відповідно до таблиці станів було створено логічну схему (рис. 2.10). Відповідно до формули визначаються три рівні напруг (V_{max}, V_{ref} та U_{min}), що визначатимуть стан технічної тривоги.

Смема підключення шлейфів до Arduino NANO зображено (рис 2.11) як підключення до схем зчитування стану шин у цій схемі внас є Шина 1 Шина 2 Шина3 Шина4. Дані зних зчитуються в квадратик схеми мікросхеми LM 248 ці блоки мікросхеми LM 248 це невставні мікросхеми а збірка ця схема є на рисунку 2.12. Кожен цей квадратик на рисунку 2.11 де написано LM 248 насправді схема 2.12 і тут внас вже поступає елемент V це вже внас поступає елемент в нас на кожну шину доводить від першої до Arduino NANO. Для Шини1 контролює стан чи непорушено схеми підключення шини а вже від A1 контролює вже безпосередньо стан шини тобто в нас на кожну шину є по 2 входи. Так само для Шини 2 внс є від A2 який контролює цей тихнічний стан шини і від A3 контролює внас вже стан безпосередньо безпеки шини так само для A4 A5. Так само для Шини 3 і A5 і A6. Для Шини 4 є A6 і A7 на цій схемі LM 248 вони використовуються входи напруг в нас є V_{max} V_{ref} V_{min}.

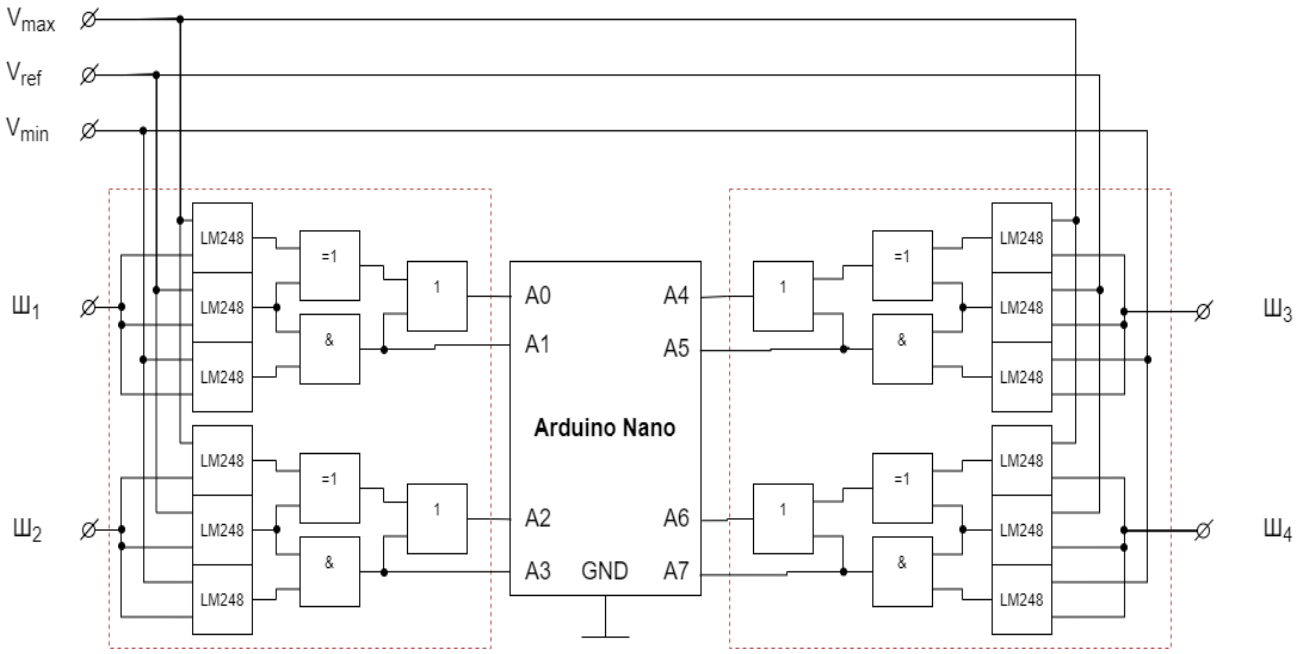


Рисунок 2.11 — Схема підключення шлейфів до Arduino NANO

Схема контролю напруги складається з: R1 — вхідний опір; R2 — опір пониження напруги для операційного підсилювача; R3 та R4 — опори під налаштування напруги спрацювання; R5 — вихідний опір, де напруга понижується до рівня логічних елементів.

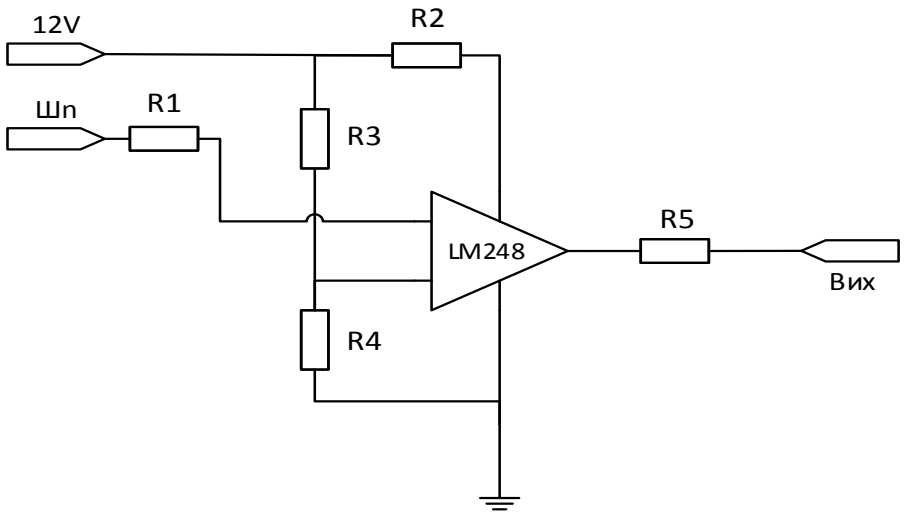


Рисунок 2.12 — Реалізація схеми контролю напруги

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Побудова мережевої архітектури

Перед безпосередньою розробкою прошивки (скетчу) контролера сигналізації та серверної компоненти слід визначити, яким чином буде відбуватись між ними взаємодія, які дані будуть передаватись та зберігатись. Найкращим варіантом при проектуванні є схеми мережевої взаємодії та схеми таблиць баз даних.

Мережева архітектура — це комбінація стандартів, топології і протоколів, які утворюють працездатну мережу.

Мережева архітектура характеризує загальну структуру мережі, тобто всі компоненти завдяки яким мережа функціонує як апаратні засоби так і системи ПЗ. Кожна мережева архітектура має свої характеристики, параметри продуктивності, апаратні та програмні засоби. Це дає проектувальнику полегшенні можливості створення мережі, яка має задовольняти вимогам функціонування. Тобто проектувальник обгрунтовує вибір мережева архітектура і не проводить внутрішнього аналізу та розрахунку мережі.

Основними складовими елементами мережної архітектури є: адаптер, протокол, клієнт, сервер.

Адаптер — це найнижчий рівень мережної архітектури, який забезпечує зв’язок між фізичним кабелем і операційною системою комп’ютера.

Протокол — це набір угод і правил, які визначають тип, фізичні сигнали, їх послідовність в часі, алгоритми прийому, контролю і передачі повідомлень, а також склад службової інформації самих повідомлень.

Сервер — як правило, є одною з визначальних ланок будь-якої локальної мережі — інструментом, який не проводити самостійні дії. Його функціональна суть полягає в тому, що серед взаємодіючих в локальній комп’ютерній мережі процесів виділяється деякий особливий процес, що називається серверним і фізично реалізований на сервері. Інші процеси називаються клієнтами [15].

Для кращого розуміння мережевих процесів було виведено основні операції, які можуть відбуватись між хмарою, контролером та користувачем:

— реєстрація користувача в хмарі;

— реєстрація пристрою користувача у хмарі;

— зміна параметрів пристрою користувача у хмарі;

— отримання повідомлень тривоги від пристрою до хмари;

— передача повідомлень користувачеві за допомогою боту.

На основі даних операцій та структурної схеми пристрою було побудовано схему мережевої взаємодії (рис. 3.1).

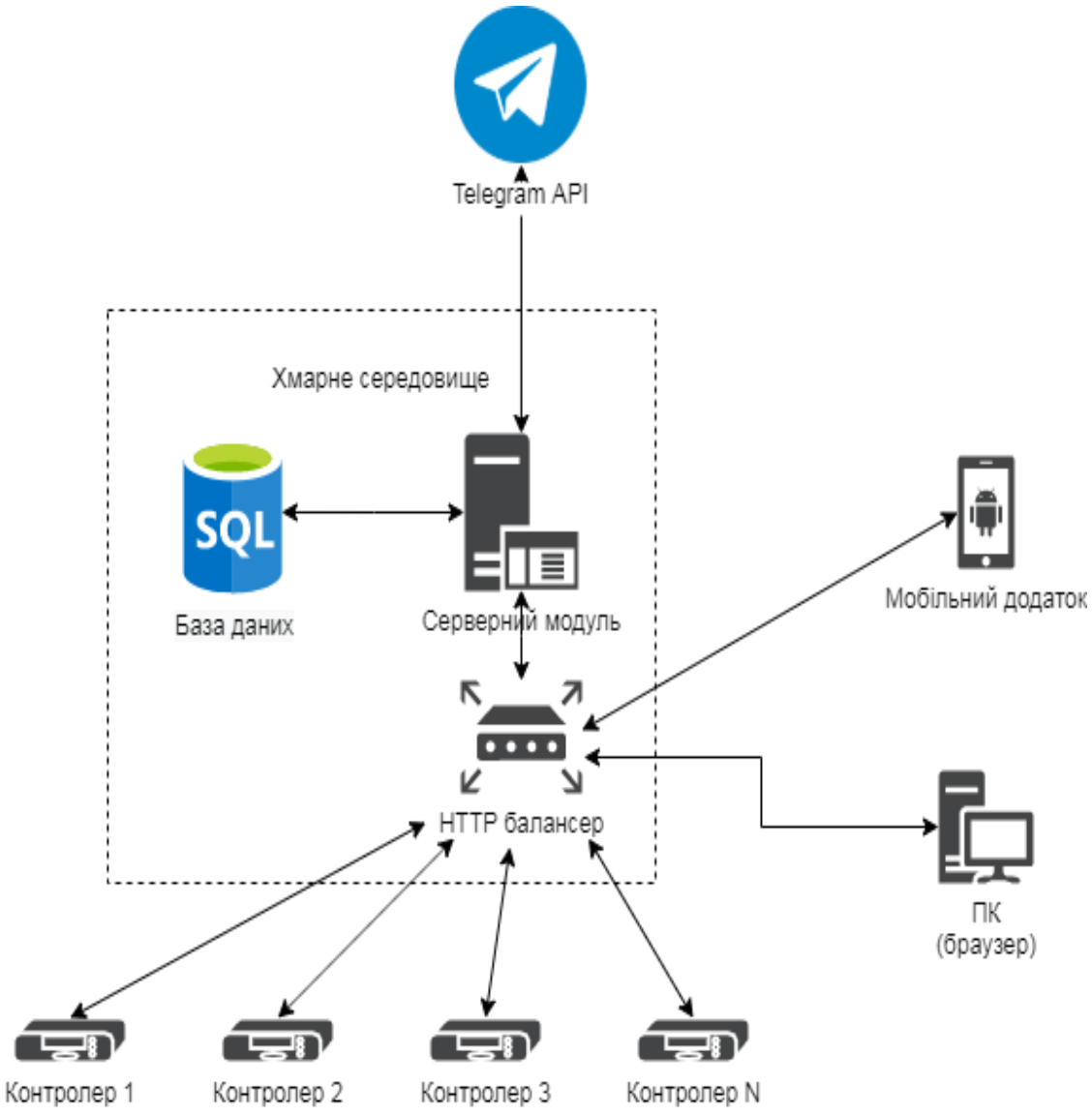


Рисунок 3.1 — Схема мережевої взаємодії

Схема на рисунку 3.1 описує як буде відбуватись підключення контролерів користувачів до хмари, через яку будуть пересилатись повідомлення в месенджери та мобільні додатки для оповіщення тривоги. Серверна компонента опрацьовуватиме всі API запити, які надходитимуть на HTTP — сервер.

3.2 Розробка мікропрограми системи безпеки

Пристрій користувача — контролер сигналізації, матиме власну програму за якою працюватиме. Така програма називається мікропрограмою. Вона працюватиме відповідно до передбаченої структури пристрою.

Мікропрограмою забезпечено виконання наступних пунктів:

— ініціалізація;

— зчитування стану шлейфів;

— обмін даними з хмарою в режимі реального часу;

— конфігурація та показ стану через браузер (по локальній мережі);

— надсилання СМС в разі потреби.

Мікропрограмного керування виробляє послідовність сигналів, необхідних для виконання програми в комп'ютері. Програма складається з деякої послідовності команд. Команда в комп'ютері виконується за один або за декілька тактів, в кожному із яких виконується одна або декілька мікрооперацій.

Кожна мікрооперація представляє собою деяку елементарну дію передачі або перетворення інформації, яка ініціалізація поступленням керуючого сигналу на вхід керування відповідного пристрою.

Послідовність елементарних мікро наказів, які пристрій керування формує в одному такті називають мікрокомандою.

Послідовність мікрокоманд, що необхідно виконати для виконання однієї команди називаються мікропрограмою. Звичайно, мікропрограма може складатися і лише з однієї мікрокоманди.

Основними принципами, які покладені в основу побудови пристрою мікро програмного керування, є наступні:

— всі мікро накази, які повинні бути виконані в одному такті роботи комп'ютера, збираються в одне керуюче слово, яке називають мікрокомандою;

— кожній команді з системи команд комп'ютера ставиться у відповідність послідовність мікрокоманд, необхідних для її виконання, тобто мікропрограма виконання команди в комп'ютері;

— всі мікрокоманди зберігаються в пам'яті — це може бути основна пам'ять комп'ютера, але в більшості комп'ютерів для зберігання мікрокоманд використовується окрема пам'ять, яку називають пам'яттю мікрокоманд;

— для реалізації деякої команди необхідно зчитати з пам'яті мікрокоманд відповідну послідовність мікрокоманд (мікропрограму) та подати розподілену в часі послідовність керуючих сигналів на відповідні керуючі входи вузлів комп'ютера.

Кроки створення мікропрограми:

— написання програми однією з мов;

— компіляція (може бути у декілька етапів);

— збірка (linking);

— запис машинного коду у ПЗП мікроконтролера.

Мікропрограма — це вміст пам’яті програм мікроконтролера, його програмне забезпечення. “Прошивкою” називають також: процес запису firmware у пам’ять програм мікроконтролера; файл з програмою, готовий для запису у пам’ять мікроконтролера.

Обмін даними з серверною компонентами у хмарі здійснюється за допомогою файлу api.php, опис розробки якого описано у підрозділі 3.2. Для обміну інформацією передбачено таких два базових режими: "getphone" (отримання пристроєм інформації з хмари) та "setobjstat" (оновлення інформації про об’єкт в базі даних хмари).

Для налаштування та моніторингу стану пристрою у локальній мережі передбачено управління через веб інтерфейс (рис 3.1), на який можна зайти,

вписавши наприклад адресу http://192.168.1.101/ — це адреса http протокола даних.

Рисунок 3.2 — Налаштування контролеру через браузер

Для налаштування контролеру через браузер було взято зразок меню CAPsMAN вкладка Configurations, що зображено на рисунку 3.3.

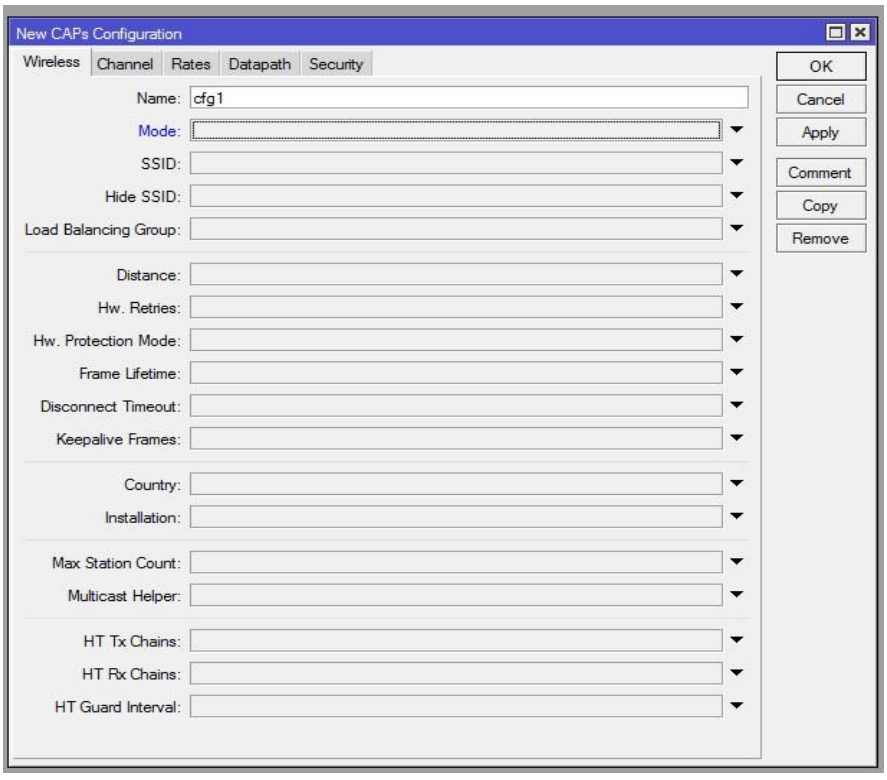


Рисунок 3.3 — Меню CAPsMAN

У вікні бачимо вкладки зі схожими назвами Channels, Rates, Datapath, Security, що зображено на рисунку 3.4.

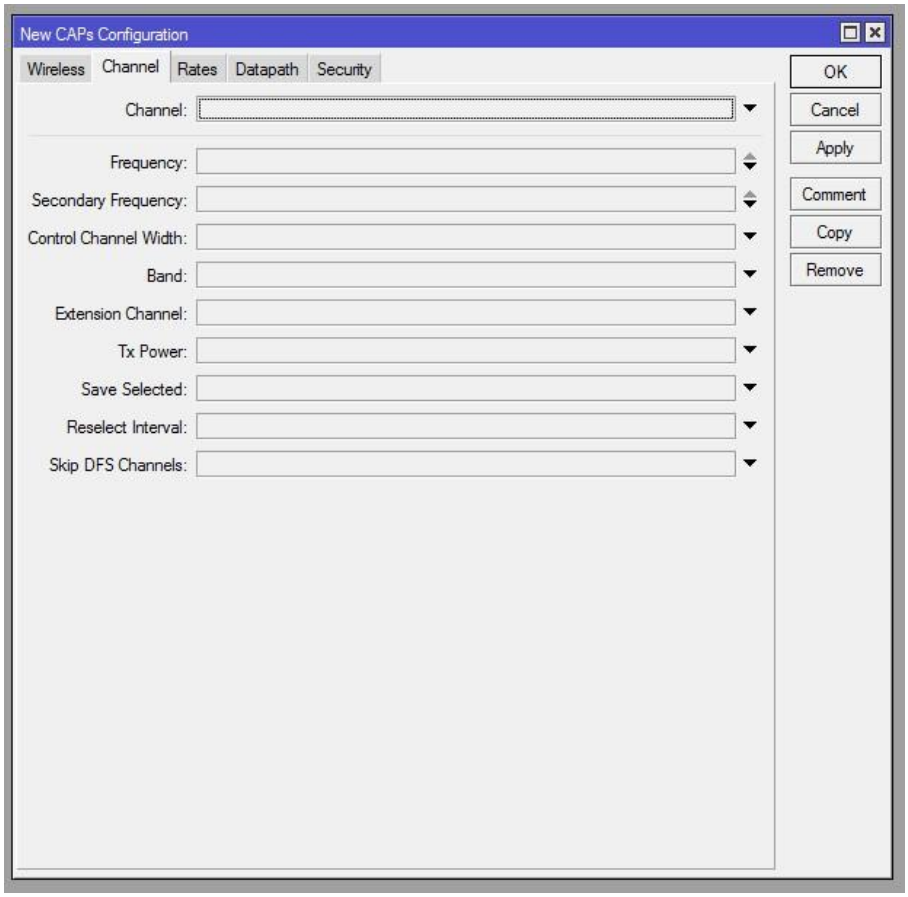


Рисунок 3.4 — Вкладка Wirewless настройка параметрів бездротової мережі.

Вкладка Wirewless містить ось такі дані:

— Name це назва профілю контролера;

— Mode це режим роботи пристрою AP;

— SSID це назва бездротової мережі;

— за допомогою Hide SSID можна зробити мережу прихованою, за потребою.

Load Balancing Group — параметр який дозволяє розподіляти підключені точки доступу. Тут потрібно вказати кількість підключень.

Distance — параметр відстеження користувачів (залишити за замовчуванням).

Hw retries — кількість повторів параметра Distance (залишити за замовчуванням).

Hw Protection Mode — захисний режим параметра Distance (залишити за замовчуванням).

Frame Lifetime — (залишити за замовчуванням).

Disconnect Timeout — відноситься до параметру Distance (залишити за замовчуванням).

Keepalive Frames — відноситься до параметру Distance (залишити за замовчуванням).

Country — тут потрібно вибрати регіон використання, після вибору в силу вступлять настройки радіо модуля для даного регіону, кількість дозволених каналів і максимальна потужність.

Installation — місце інсталяції точки доступу.

Max Station Count — максимальна кількість осіб на бездротову мережу (за замовчуванням 2007 підключень).

Multicast Helper — управління мультикаст потоками в бездротовій мережі.

HT Tx chain — які антени використовувати для передачі.

HT Rx chain — які антени використовувати для прийому.

HT Guard Interval — чи дозволити використання короткого захисного інтервалу «any», буде використовувати short або long, в залежності від швидкості передачі даних, «long» буде використовувати тільки long.

3.3 Розробка серверної компоненти

В сучасному світі неможливо обійтися без використання різного роду систем комунікації. Вони стали невід’ємною частиною нашого життя. Майже кожна сучасна людина користується тою чи іншою системою обміну інформації з іншими людьми. Частіше всього для обміну інформацією використовуються

мобільні платформи. В переважній більшості це такі платформи як Android, iOS та Windows Phone. Інформація, яка передається від одного користувача до іншого, часто може бути конфіденційною або з обмеженим доступом. Для цього мають використовуватися відповідні методи захисту інформації. Сьогодні увага дослідників інформаційної безпеки зосереджена на покращенні роботи криптографічних алгоритмів та примітивів, на яких вони засновані. Тим не менш, питання побудови комплексних систем захисту в системах обміну миттєвими повідомленнями залишаються розглянутими недостатньо. Дана стаття присвячена розробці комплексної системи захисту застосунку для обміну миттєвими повідомленнями. Будь який застосунок для роботи з миттєвими повідомленнями складається з трьох основних частин. Першою складовою частиною є програмний код, виконуваний на мобільній платформі. Другою частиною є канал зв’язку. Останньою частиною є програмний код, виконуваний на стороні серверу. В даній статті розглянуто кожну з основних частин для побудови комплексної системи захисту застосунку для обміну миттєвими повідомленнями. Обґрунтовано вибір тих чи інших компонентів для побудови системи захисту, приведено схеми роботи програмного коду виконуваного на мобільній платформі Android, обґрунтовано вибір форм передачі користувацької інформації через канал зв’язку. Надано рекомендації по вибору каналу зв’язку та забезпеченню його безпеки. Також приведено схеми роботи серверної сторони застосунку. Обґрунтовано вибір тих чи інших хеш-функцій, функцій перевірок вхідної інформації, підходів до реалізації системи захисту, системи зберігання і обробки користувацької інформації на стороні серверу. У застосунку також продемонстровано деякі недоліки поточної версії мобільної платформи Android з точки зору інформаційної безпеки. Показано, як зловмисники можуть використати ці недоліки, та надано поради по уникненню їх експлуатації зловмисником. Результати роботи можуть бути використані для захисту інформації у подібних застосунках.

Серверний компонент являє собою набір скриптів та web сторінок, створеними за допомогою технологій HTML, PHP та MYSQL.

Ядром комплексу серверний компоненту слугує база даних, де зберігається інформація про користувачів, пристрої та стани їх тривоги на окремих об'єктах (шлейфах).

Для зручності призначення окремих скриптів, створено таблицю 3.1

Таблиця 3.1 Окремі скрипти та їх призначення

Назва скрипта	опис
INDEX.HTML	Показує навігацію по серверній компоненті. Є головним файлом
LOGIN.PHP	Наступна сторінка, на якій можна ввійти в систему під певним користувачем
STATUS.PHP	Показує перелік всіх пристроїв та стан кожного з них
OPTIONS.PHP	Дозволяє змінювати налаштування кожного пристрою та його об’єктів.
USER.PHP	Налаштування користувача, де можна змінити номер та налаштувати бот для телеграм
API.PHP	Опрацювання запитів від Arduino контролера

Серверна компонента має гарний та інтуїтивно зрозумілий інтерфейс, який легко налаштовувати. Кожна сторінка має свій окремий дизайн: сторінка входу використовує форму з трьома HTML елементами типу INPUT.

Дизайн сторінки входу у систему наведено на рисунку 3.5.

Для головного меню було визначено більш досконалий дизайн, який складається з заголовку, блоку TABLE (таблиця, де виводяться всі контролери, які

прив’язав до свого телеграм боту користувач), а також одне посилання переходу на сторінку детальних опцій (рис. 3.5).

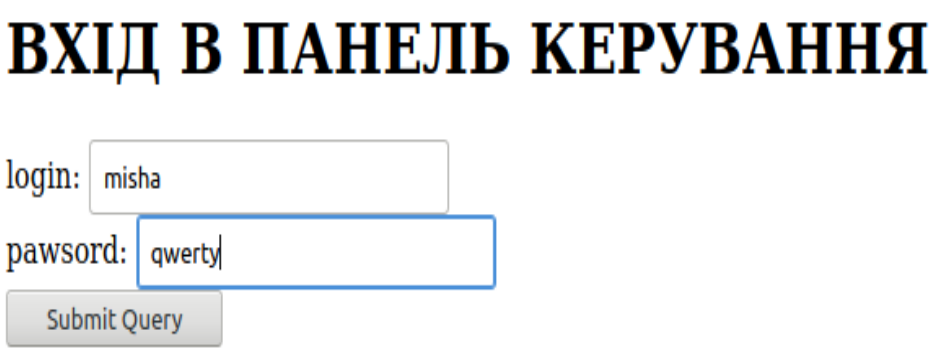


Рисунок 3.5 — Дизайн сторінки входу у систему

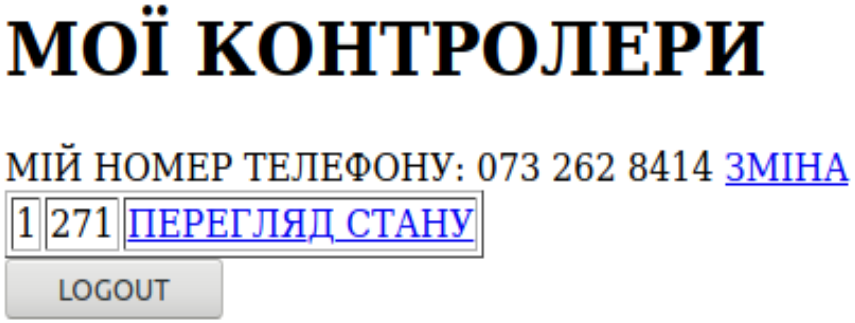


Рисунок 3.6 — Дизайн головного меню серверної компоненти

Взаємодію між мікропрограмою Arduino та серверною компонентами забезпечено на за допомогою засобів HTTP REST.

Код скетчу Arduino наведено у додатку Г. У додатку Д наведено код установки структури для серверної компоненти в хмарі [19].

Скрипт API.PHP працює на базі бібліотеки Restbox і її класу RestUtil. Тут вона підключена задопомогою autoload.php з інструментарію Compozer. Скрипт API.PHP може працювати у трьох режимах він може працювати безпосередньо з контролерами за контролера відповідає клас device може працюва

з мобільними клієнтами під мобільні клієнти існує клас Mobale і також ще є клас Telegram працює із ботом і дозволяє опрацьовати його запити. Дальше розберемо окремо по методам методи виконуються в функції API Royter в який відбувається виклик цих методів.

Метод device_auth зображено на рисунку 3.7 він дозволяє задопомогою метода outs створити сесію для роботи пристрою пристрій це контролер на Arduino він підключається і заносить свій індифікатор та токен доступу. В результаті він отримує індифікатор сесії у випадку якщо він не пост а запит або в нас отримує неправильні дані то нас видає помилка 400.

```
// Device methods
case 'device_auth':
      if($context->method == "POST") {
      $sess_data = Device::CreateSession(
            $context->post['device_id'],
            $context->post['token']
      )
      return RestUtil::ReturnJson( $sess_data );
      } else {
      return RestUtil::ReturnError(400, 'Bad request');
      }
      break;
```

Рисунок 3.7 — Метод device_auth

Наступний метод в нс для оновлення стану пристроїв device_update_status зображено на рисунку 3.8 він також працює в режимі пост запитів і приймає дані токини сесії. Як результат крім токену вказується індифікатор пристрою та його токен доступу в результаті ведеться про опрацювання цього запиту де і безпосередньо оновлюється дані про стан пристрою і некритичні події і зміни на пристрою.

```
case 'device_update_status':
      if( $context->method == "POST" &&
      Device::IsSessionValid(
            $context->post['token']
      )
      ) {
      $result = Device::SetStatus(
            $context->post['device_id'],
            $context->post['token']
      )
      return RestUtil::ReturnJson( $result );
      } else {
      return RestUtil::ReturnError(400, 'Bad session');
      }
      break;
```

Рисунок 3.8 — Метод device_update_status

Останім методом API.PHP для роботи з пристроями в нас є метод device_alarm зображено на рисунку 3.9 цей метод також зчитує дані із методу пост зчитується токен доступу пристрою його індифікатор також передається objik це індифікатор об’єкта і його стан метод device_alarm призначений для передачі критичних повідомлень наприклад простан шлейфів і про тривогу. Резльтаті його виконання видається код наскільки успішно був виконаний запит що вхідні дані неправильні додається також помилка 400.

```
case 'device_alarm':
      if( $context->method == "POST" &&
      Device::IsSessionValid(
            $context->post['token']
      )
      ) {
      $result = Device::Alarm(
            $context->post['device_id'],
            $context->post['object'],
            $context->post['status']
      )
      return RestUtil::ReturnJson( $result );
      } else {
      return RestUtil::ReturnError(400, 'Bad session');
      }
      break;
```

Рисунок 3.9 — Метод device_alarm

Наступні методи розглянуто mobile_auth зображено на рисунку 3.10 які опрацьовають запити від додатків перший метод це афтифікація мобільного додатку через метод POST. Також передається індифікатор пристрою його токен єдина відміність мобільного пристрою і контролера то що контролер натсилає сповіщення на сервер а мобільний пристрій вже отримує безпосередньо до них доступ але в цілому процес афтифікації схожим.

```
// Mobile methods
case 'mobile_auth':
      if($context->method == "POST") {
      $sess_data = Mobile::CreateSession(
            $context->post['device_id'],
            $context->post['token']
      )
      return RestUtil::ReturnJson( $sess_data );
      } else {
      return RestUtil::ReturnError(400, 'Bad request');
      }
      break;
```

Рисунок 3.10 — Метод mobile_auth

Наступний метод який описує в нас доступ мобільних пристроїв до серверної компоненти є метод mobile_list_ device зображено на рисунку 3.11 Цей метод дозволяє отримати перелік контролерів підєднані до акаунту користувача і таким чином відбувається індифікатору користувача їх перерахунок і також надаються відповіді на дані про стан пристроїв. Якщо внас пристрій невідомий або є помилки в сесії тоді буде помилка 400.

```
case 'mobile_list_devices':
      if( Mobile::IsSessionValid(
      $context->post['token']
      ) ) {
      $result = Device::List(
            Mobile::GetUserID(),
      )
      return RestUtil::ReturnJson( $result );
      } else {
      return RestUtil::ReturnError(400, 'Bad session');
      }
      break;
```

Рисунок 3.11 — Метод mobile_list_ device

В нас є два метода які дозволяють користувачам мобільного додатків дізнатись стан пристроїв з які вони належать тим чи іншим користувача. Перший з методів яких дозволяє користувачем мобільним пристроєм дізнатися про тривогу це є mobile_list_ alarms зображено на рисунку 3.12 даний метод сповіщає про всі останні події які трапилися на які потрібно реагувати. В результаті тут вони в нас отримується індифікатор користувача всі в нас з класу device створюється перелік всіх тривог які призначені для конкретного користувача в незалежності від присторої де він виникнув це все відправляється назад в клієнт.

Другий метод нам дозволяє працювати з тривогами це є метод mobile_mute_ alarms зображено на рисунку 3.12 Цей метод також верифікує сесію і таким чином отримується індифікатор користувача який саме пристрій тут в нас задіяний відповідно тривоги які знього надійшли вони позначаються як ті що користувач з ними ознайомився і відповідно вони повторно вже небудуть відображатися у веб інтерфейсі і мобільному інтерфейсі і месенджерах які потребують уваги.

```
case 'mobile_list_alarms':
      if( Mobile::IsSessionValid(
      $context->post['token']
      ) ) {
      $result = Device::ListAlarms(
            Mobile::GetUserID(),
      )
      return RestUtil::ReturnJson( $result );
      } else {
      return RestUtil::ReturnError(400, 'Bad session');
      }
      break;

case 'mobile_mute_alarms':
      if( Mobile::IsSessionValid(
      $context->post['token']
      ) ) {
      $result = Device::MuteAlarm(
            Mobile::GetUserID(),
            $context->post['device_id'],
      )
      return RestUtil::ReturnJson( $result );
      } else {
      return RestUtil::ReturnError(400, 'Bad session');
      }
      break;
```

Рисунок 3.12 — Методи mobile_list_ alarms і mobile_mute_ alarm

В нас також є клас Telegram останій він також має три методи одни для афтифікації інший для перегляду даних. Перший метод це mess_auth зображено на рисунку 3.13 він дозволяє в нас задопомогою методу POST отримати ідифікатор чату у Telegram а також токен доступу який користувач отримав задопомогою посилання. В результаті видаються всі дані які необхідні для роботи бота.

```
// Telegram bot
case 'mess_auth':
        if($context->method == "POST") {
        $sess_data = Telegram::CreateSession(
                $context->post['chat_id'],
                $context->post['token']
        )
        return RestUtil::ReturnJson( $sess_data );
        } else {
        return RestUtil::ReturnError(400, 'Bad request');
        }
        break;
```

Рисунок 3.13 — Метод mess_auth

Другий метод mess_list_device зображено на рисунку 3.14 який працює на базі класу Telegram він перевіряє чи дійсна сесія і достає з нього індифікатор користувача і у відповідь надсилається в нас перелік всіх пристроїв і короткий опис їх стану і таким чином задопомогою натиснення кнопки у боті що передбачає заведення всіх пристроїв відбувається виведення їх стану і зокрема як і повідомлення і також детальну інформацію по кожному пристрою.

```
case 'mess_list_devices':
        if( Telegram::IsSessionValid(
        $context->post['token']
        ) ) {
        $result = Device::List(
                Telegram::GetUserID(),
        )
        return RestUtil::ReturnJson( $result );
        } else {
           return RestUtil::ReturnError(400, 'Bad session');
        }
        break;
```

Рисунок 3.14 — Метод mess_ list_devices

Останій API метод який внас призначений для роботи з ботом є mess_ list_ alarms зображено на рисунку 3.15 цей метод нам дозволяє по токену автифікації отримати перелік всіх тривог і таким чином надіслати їх в чат бот перевіряє чи є тривоги і без посередньо надсилає їх в чат користувача. Тут пересилається лише повідомлення про тривогу навідміно від веб інтирфейсу і мобільного додатку нам немає можливості підтвердити можливості чи користувач ознайомився з подією чи ні. Тому що бот є інформативна функція однак користувач переконується про дані і подію або в мобільному додатку або ще у браузері.

```
case 'mess_list_alarms':
        if( Telegram::IsSessionValid(
        $context->post['token']
        ) ) {
        $result = Device::ListAlarms(
              Telegram::GetUserID(),
              true
        )
        return RestUtil::ReturnJson( $result );
        } else {
        return RestUtil::ReturnError(400, 'Bad session');
        }
        break;
```

Рисунок 3.15 — Метод mess_ list_ alarms

ВИСНОВКИ

В ході роботи над проектом розглянуто існуючі аналоги та їх конкурентні переваги. Запропоновано варіанти здешевлення компонентної бази пристрою, а також шляхи забезпечення багатоканального оповіщення. Застосування результатів дослідження дозволить використовувати більш дешеві радіо компоненти замість дорого вартісних контролерів.

Розроблено пристрій на основі контролера Arduino Nano, мікропрограму для нього, а також серверну компоненту, що забезпечує багатоканальне оповіщення. Розроблена система має такий функціонал:

— відслідковування стану підключених датчиків;

— синхронізація налаштувань з хмарним середовищем;

— сповіщення тривоги за допомогою СМС повідомлень;

— здійснення багатоканального оповіщення з використанням месенджерів та миттєвих сповіщень;

— можливість розширення системи завдяки підключенню бездротових модулів.

Мікропрограму для контролеру Arduino Nano написано мовою програмування C/C++, з використанням середовища розробки програмного забезпечення Arduino IDE. Серверну компоненту написано на мові програмування PHP з використанням діалекту запитів до бази даних MySQL. Розроблений веб інтерфейс є зручним та зрозумілим.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Борботько Т.В., Линьков Л.М., Маліков В.В. Захист об'єктів різних форм власності від несанкціонованого доступу. Мінськ: Поліграфічний центр., 2008. 187с.

2. Груба І.І. Система охоронної сигналізації технічні засоби виявлення. Харків: Солон прес. Україна, 2011. 220с.

3. Міжнародний кодекс із систем пожежної безпеки (резолюція MSC.9873) Видавництво: ЦНИИМФ Рік видання 2016 184с.

4. ADT Orion 8K країна виробник Україна: розробник і виробник компанія Вінниця ТОВ «Тірас 12».

5. GSM Kerui G18 країна виробник Китай: розробник і виробник компанія «Kerui».

6. ATIS Kit 200T країна виробник Україна: розробник і виробник компанія Вінниця «Atis».

7. Оригінальним автором програми Wireshark Джеральд Комбс 1998 році: електронний ресурс URL–https://www.wireshark.org/about.html#authors.

8. Оригінальним автором програми Proteus Design Джоном Джеймсоном заснованоу1998 році: електронний ресурс URL– https://www.labcenter.com/about.

9. Засновниками програми Arduino IDE є п’ятеро друзів Дэвид Куартилльз, Джанлука Мартино, Том Иго , Дэвид Мелис , и Массимо Банци заснованоу 2005 році: електронний ресурс URL– https://www.arduino.cc.

10. Кушнір Л.М. Основи автоматики та цифрової техніки. Логічні елементи та комбінаційні пристрої: навчальний посібник для студентів.

11.Мікроконтролерна система.: електронний ресурс– https://en.wikipedia.org/wiki/Microcontroller.

12. Плата Arduino Nano v 3.0 розписування схеми: електронний ресурс URL – https://arduinomaster.ru/platy-arduino/plata-arduino-nano.

13. Магауенов Р.Г. Системи охоронної сигналізації: основи теорії і принципи побудови. Навчальний посібник для вузів. 2-вид. М.: Гаряча лінія – Телеком. 2008. 496 с.

14. Операційні підсилювачі ─ конструкція і застосування автор Л.П.Хуельсман видавництво Макгроу-Хілл.

15. Ільчук А.В. Технології комп’ютерної безпеки. Монографія 2. МЕГУ, Рівне, 2011.-97 с.

16. Навчальне видання автор Гайна Георгій Анатолійович навчальний посібник «Основи проектування баз даних».

17. Хомоненко О.Д., Циганков В.М., Мальцев М.Г. Бази Даних і їх розвиток ─ СПб: Корона 2004. ─ 736 с.

18. Майданюк В.П. Рибак А.О. Крос платформний додаток для доступу до бази даних, матеріали міжнародної науково практичної інтернет конференції Молодь в технічних науках: дослідження, проблеми, перспективи. URL:http://conf.inmad.vntu.edu.ua/fm/index.php?page=materials&line=29&mat

19. Майданюк В.П., Пєтух А.М. Інтерфейс "Користувач комп'ютер": навчальний посібник. Вінниця: ВДТУ.

20. Магауенов Р.Г. Системи охоронної сигналізації: основи теорії і принципи побудови. Навчальний посібник для вузів. 2-вид. М.: Гаряча лінія – Телеком. 2008. 496 с.

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ ВНТУ
д.т.н., проф.
_____О. Д. Азаров
«29» квітня 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської дипломної роботи
«Одномодульна мультисенсорна системп безпеки»
08-23.БДР.042.00.000 ПЗ

Керівник роботи к.т.н. доц. каф. ОТ
_____ Томчук М.А.

Студент групи 1КІ–20мс
_____ Ярошевський М.М.

ВНТУ 2022

1 Підстави для виконання бакалаврської дипломної роботи (БДР) наступні:

— актуальність розробки, яка полягає у вирішенні проблеми швидкого доступу до тестування студентів, і їх подальшого самотестування, які дозволять отримати додаток для вирішення проблеми, на снові якого буде можливість швидко пройти тест і пройти самотестування для впевненості студентів в своїх знаннях;

— наказ про затвердження теми бакалаврської дипломної роботи.

2 Мета і призначення БДР наступні:

— метою БДР є розробка розробка пристрою сигналізації для спостереження за станом спостережуваних об’єктів з передачею змін в хмарне середовище;

— призначення розробки полягає у виконанні комплексної бакалаврської диплоної роботи із подальшим розвитком.

3 Джерела розробки:

— контент для наповнення охороної системи безпеки;

— інформація про технології розробки систем безпеки;

— використанням GSM мереж і також WiFi і хмарного середовища.

4 Вимога до виконання БДР наступна:

— створення пристрою універсального із доступним бюджетом який непоступається нічим своїм аналогам і навіть кращий в деяких характеристиках;

— статичні і динамічні сторінки.

5 Етапи БДР та очікувані результати приведено в таблиці А.1.

6 Матеріали, що подаються до захисту БДР: пояснювальна записка БДР, графічні і ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відзив наукового керівника, рецензія опонента, анотації до БДР українською та іноземною мовами, нормоконтроль про відповідність оформлення БДР діючим вимогам.

Таблиця А.1 — Етапи виконання роботи

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання		Примітка
		початок	закінчення	
1	Постановка задачі роботи	09.01.2022		Вступ
2	Пошук матеріалів по технологіям розробки на мові C++ і PHP	10.02.2022	25.02.22	Розділ 1
3	Аналіз та вибір технології для роботи із базою даних	26.02.22	28.02.22	Розділ 1
4	Обгрунтування та вибір засобів реалізації структури схеми пристрою і алгоритму роботи програми.	29.02.22	06.03.22	Розділ 2
5	Проектування програмного забеспечення для пристрою	07.03.22	25.04.22	Розділ 3
6	Розробка програмного забеспення пристрою	26.04.22	20.05.2022	Розділ 4
7	Оформлення пояснювальної записки та ілюстративного матеріалу	21.05.2022	04.06.2022	Пояснювальна записки і додатки
8	Перевірка якості виконання бакалаврської роботи та усунення недоліків	07.06.2022		Презентація

7 Виконання етапів графічної та розрахункової документації БДР контролюється науковим керівником згідно зі встановленими термінами. Захист БДР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення БДР викладені в:

— методичних вказівках до дипломного проектування;

— ДСТУ_3008 : 2015 «Звіти в сфері нацки і техніки. Структура і правила оформлення»;

— ДСТУ_8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104 – 2006 «Єдина система конструкторного локументації. Основні написи».

Технічне завдання до виконання отримав _____Ярошевський М.М

ДОДАТОК Б

Структурна схема пристрою

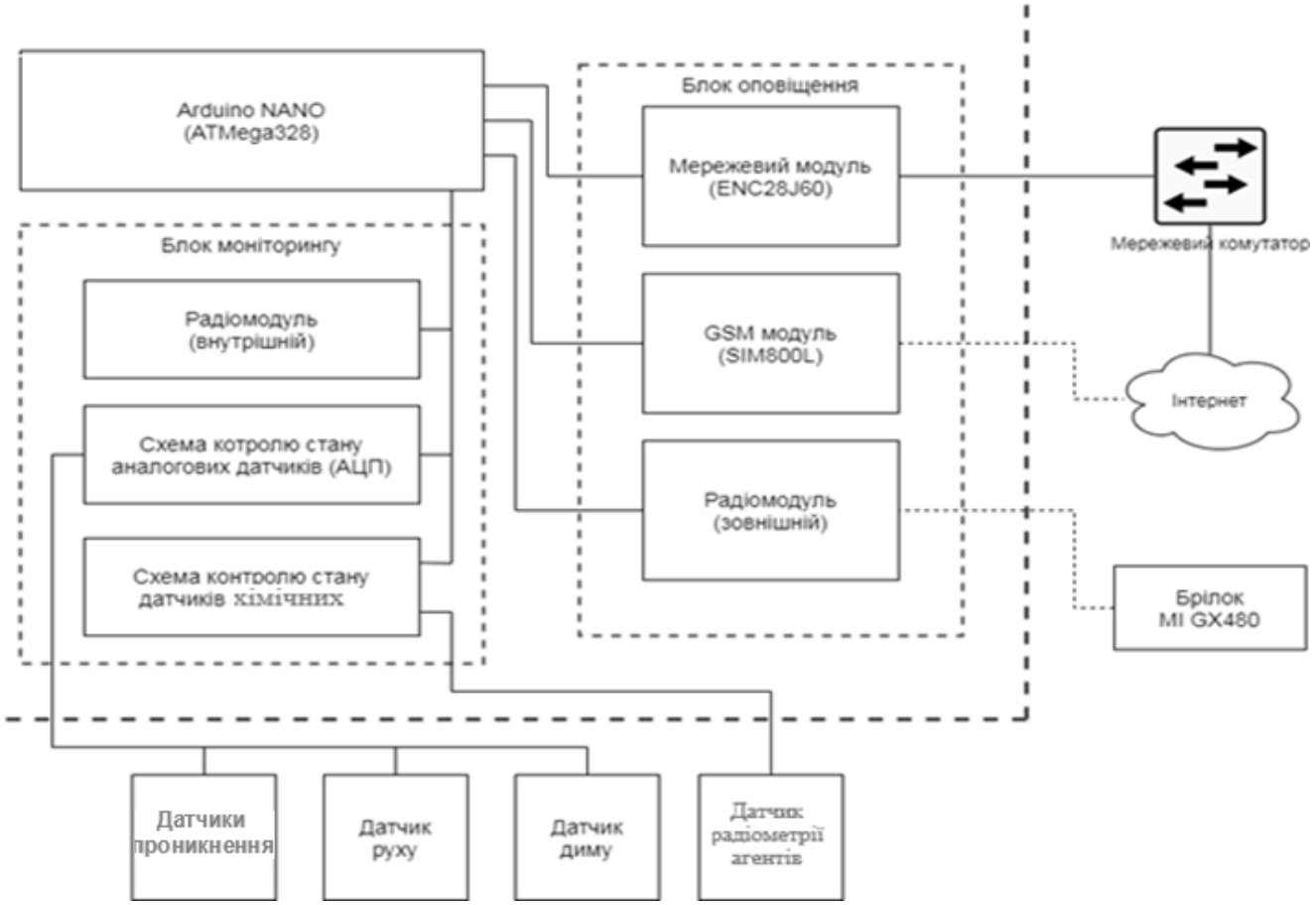


Рисунок Б.1 — Структурна схема системи безпеки

ДОДАТОК В

Блок-схема алгоритм роботи програми

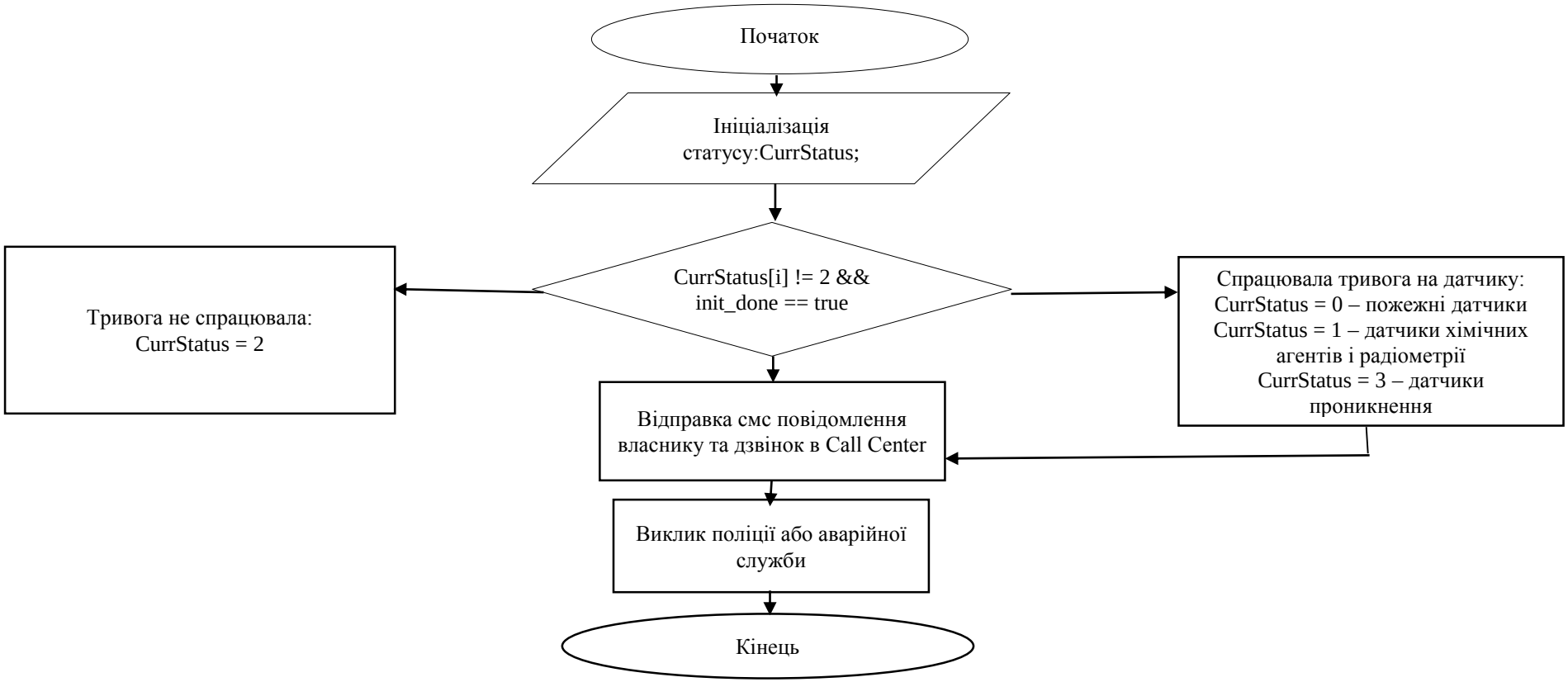


Рисунок В.1 — Блок-схема алгоритм роботи програми

Додаток Г

Лістинг файла signal_controller.ino

```
#include <EtherCard.h>
#include <SmartSMS.h>
byte Ethernet::buffer[900];
BufferFiller bfill;
char *userphone;
bool init_done;
const char website[] PROGMEM = "lab.vntu.org";
const char app_pref[] PROGMEM = "/tomchuk68/";
const char dev_serial[] PROGMEM = "27f11cd8-4dd6-11ea-9358-0f62c4ccd782";
static byte mymac[] = {
  0x27,0xf1,0x1c,0xd8,0x4d,0xd6
};
static byte myip[] = {
  10,90,90,90
};
int StatusPins[] = {
  1,3,5,7
};
int ErrorPins[] = {
  2,4,6,8
};
int StatusHistory[] = {
  0,0,0,0
};
char *StatusDesc[] = {
```

```
  "Line destroy",
  "Invasion",
  "OK",
  "Short zanykanie"
};
const char http_OK[] PROGMEM =
"HTTP/1.0 200 OK\r\n"
"Content-Type: text/html\r\n"
"Pragma: no-cache\r\n\r\n";
const char http_Found[] PROGMEM =
"HTTP/1.0 302 Found\r\n"
"Location: /\r\n\r\n";
const char http_Unauthorized[] PROGMEM =
"HTTP/1.0 401 Unauthorized\r\n"
"Content-Type: text/html\r\n\r\n"
"<h1>401 Unauthorized</h1>";
// ---------------------------------
int GetObjStatus(int pstat, int perr) {
  int val_st = 0;
  int val_er = 0;
  val_st = digitalRead(pstat);
  val_er = digitalRead(perr);
  if(val_st == 0 && val_er == 0)
    return 0;
  if(val_st == 0 && val_er == 1)
    return 1;
  if(val_st == 1 && val_er == 0)
    return 2;
  if(val_st == 1 && val_er == 1)
```

```
    return 3;
  return 0;
}
bool PrintObjStatus(int pstat, int perr) {
  int state = 0;
  state = GetObjStatus(pstat, perr);
  if(state == 2) return true;
  else return false;
}
static void set_userphone (byte status, word off, word len) {
  userphone = (char *) Ethernet::buffer + off;
}
void index_html() {
  bfill.emit_p(PSTR("$F"
    "<title>Objects status</title>"
    "<h1>Objects status</h1>"
    "Port 1: $F<br />"
    "Port 2: $F<br />"
    "Port 3: $F<br />"
    "Port 4: $F"),
  http_OK,
  PrintObjStatus(StatusPins[0],
ErrorPins[0])?PSTR("<font
color=\"green\"><b>PROTECT</b></font>"):PSTR("<font
color=\"red\">FAIL</font>"),
  PrintObjStatus(StatusPins[1],
ErrorPins[1])?PSTR("<font
color=\"green\"><b>PROTECT</b></font>"):PSTR("<font
color=\"red\">FAIL</font>"),
```

```
  PrintObjStatus(StatusPins[2],
ErrorPins[2])?PSTR("<font
color=\"green\"><b>PROTECT</b></font>"):PSTR("<font
color=\"red\">FAIL</font>"),
  PrintObjStatus(StatusPins[3],
ErrorPins[3])?PSTR("<font
color=\"green\"><b>PROTECT</b></font>"):PSTR("<font
color=\"red\">FAIL</font>"));
}
static void def_callback (byte status, word off, word len) {
  return;
}
// ----------------------------------
void setup() {
  Serial.begin(9600);
  if (ether.begin(sizeof Ethernet::buffer, mymac,10) == 0);
  if (!ether.dhcpSetup()) {
    ether.staticSetup(myip);
  }
  ether.printIp("Device IP: ", ether.myip);
  for (int i=0; i <= 3; i++){
    pinMode(StatusPins[i], INPUT);
    pinMode(ErrorPins[i], INPUT);
  }
  if (!ether.dnsLookup(website))
    Serial.println("DNS failed");
  ether.printIp("Server: ", ether.hisip);
  char *apiuri;
  sprintf(apiuri,"api.php?mode=getphone&serial=%s",dev_serial);
```

```
  ether.browseUrl(app_pref, apiuri, website, set_userphone);
  init_done = true;
}
void loop() {
  delay(1);
  word len = ether.packetReceive();
  word pos = ether.packetLoop(len);
  if (pos) {
    bfill = ether.tcpOffset();
    char *data = (char *) Ethernet::buffer + pos;
    if (strncmp("GET /", data, 5) != 0) {
      bfill.emit_p(http_Unauthorized);
    }
    else {
      data += 5;
      if (data[0] == ' ') {
        index_html();
      }
      else {
        bfill.emit_p(http_Unauthorized);
      }
    }
    ether.httpServerReply(bfill.position());
  }
  int CurrStatus[] = { 0,0,0,0 };
  for (int i=0; i <= 3; i++) {
    char *apiuri;
    CurrStatus[i] = GetObjStatus(StatusPins[0], ErrorPins[0]);
```

```
    sprintf(apiuri,                "api.php?mode=setobjstat&serial=%s&obj=%i&status=%i",
dev_serial, i, CurrStatus[i]);
    ether.browseUrl(app_pref, apiuri, website, def_callback);
   }
  for (int i=0; i <= 3; i++) {
    if(CurrStatus[i] != StatusHistory[i]) {
      if(CurrStatus[i] != 2 && init_done == true) {
        char *smstxt;
        sprintf(smstxt, "Dorogoy abonent, na objeke %i voznikla trevoga:\r\n%s", i,
StatusDesc[CurrStatus[i]]);
        SmartSMS.begin();
        SmartSMS.send(userphone, smstxt, null);
        Serial.println(smstxt);
       }
      StatusHistory[i] = CurrStatus[i];

    }
```

ДОДАТОК Д

Лістинг файл sql_schema.sql

```
CREATE SCHEMA IF NOT EXISTS `signal_server` DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci ;
USE `signal_server` ;
CREATE TABLE IF NOT EXISTS `signal_server`.`users` (
  `user_id` INT NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(255) NULL,
  `email` VARCHAR(255) NULL,
  `hash` VARCHAR(32) NULL,
  `signup_dt` DATETIME NULL,
  `login_dt` DATETIME NULL,
  `active` INT NOT NULL DEFAULT 0,
  PRIMARY KEY (`user_id`),
  UNIQUE INDEX `id_UNIQUE` (`user_id` ASC),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC))
ENGINE = InnoDB;
CREATE TABLE IF NOT EXISTS `signal_server`.`devices` (
  `device_id` INT NOT NULL AUTO_INCREMENT,
  `serial` VARCHAR(32) NULL,
  `owner_id` INT NULL DEFAULT 0,
  `active` INT NULL,
  `last_online` DATETIME NULL,
  PRIMARY KEY (`device_id`),
  UNIQUE INDEX `id_UNIQUE` (`device_id` ASC),
  INDEX `user_id_idx` (`owner_id` ASC),
  CONSTRAINT `user_id`
    FOREIGN KEY (`owner_id`)
```

```
    REFERENCES `signal_server`.`users` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
CREATE TABLE IF NOT EXISTS `signal_server`.`objects` (
  `object_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `parent_id` INT NULL,
  `port_num` INT NULL,
  `type` INT NULL,
  PRIMARY KEY (`object_id`),
  INDEX `device_id_idx` (`parent_id` ASC),
  CONSTRAINT `device_id`
    FOREIGN KEY (`parent_id`)
    REFERENCES `signal_server`.`devices` (`device_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
CREATE TABLE IF NOT EXISTS `signal_server`.`obj_status` (
  `update_id` INT NOT NULL AUTO_INCREMENT,
  `parent_object` INT NULL,
  `status_code` INT NULL,
  PRIMARY KEY (`update_id`),
  UNIQUE INDEX `update_id_UNIQUE` (`update_id` ASC),
  INDEX `parent_object_idx` (`parent_object` ASC),
  CONSTRAINT `parent_object`
    FOREIGN KEY (`parent_object`)
    REFERENCES `signal_server`.`objects` (`parent_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

ДОДАТОК Е

Лістинг файл API.PHP

```
use RestBox\RestUtil;
require __DIR__ . '/vendor/autoload.php';
require __DIR__ . '/api/device.php';
require __DIR__ . '/api/mobile.php';
require __DIR__ . '/api/telegram.php';
function API_Router($context) {
switch ($context->get['metod']) {
    // Device methods
    case 'device_auth':
        if($context->method == "POST") {
            $sess_data = Device::CreateSession(
                $context->post['device_id'],
                $context->post['token']
            return RestUtil::ReturnJson( $sess_data );
        } else {
            return RestUtil::ReturnError(400, 'Bad request');
        }
        break;
    case 'device_update_status':
        if( $context->method == "POST" &&
            Device::IsSessionValid(
                $context->post['token']
         {
            $result = Device::SetStatus(
                $context->post['device_id'],
                $context->post['token']
```

```
            return RestUtil::ReturnJson( $result );
        } else {
            return RestUtil::ReturnError(400, 'Bad session');
        }
        break;
    case 'device_alarm':
        if( $context->method == "POST" &&
            Device::IsSessionValid(
                $context->post['token']
            )
        ) {
            $result = Device::Alarm(
                $context->post['device_id'],
                $context->post['object'],
                $context->post['status']
            )
            return RestUtil::ReturnJson( $result );
        } else {
            return RestUtil::ReturnError(400, 'Bad session');
        }
        break;
    // Mobile methods
    case 'mobile_auth':
        if($context->method == "POST") {
            $sess_data = Mobile::CreateSession(
                $context->post['device_id'],
                $context->post['token']
            )
            return RestUtil::ReturnJson( $sess_data );
```

```
    } else {
        return RestUtil::ReturnError(400, 'Bad request');
    }
    break;
case 'mobile_list_devices':
    if( Mobile::IsSessionValid(
        $context->post['token']
    ) ) {
        $result = Device::List(
            Mobile::GetUserID(),
        )
        return RestUtil::ReturnJson( $result );
    } else {
        return RestUtil::ReturnError(400, 'Bad session');
    }
    break;
case 'mobile_list_alarms':
    if( Mobile::IsSessionValid(
        $context->post['token']
    ) ) {
        $result = Device::ListAlarms(
            Mobile::GetUserID(),
        )
        return RestUtil::ReturnJson( $result );
    } else {
        return RestUtil::ReturnError(400, 'Bad session');
    }
    break;
case 'mobile_mute_alarms':
```

```
    if( Mobile::IsSessionValid(
        $context->post['token']
    ) ) {
        $result = Device::MuteAlarm(
            Mobile::GetUserID(),
            $context->post['device_id'],
        )
        return RestUtil::ReturnJson( $result );
    } else {
        return RestUtil::ReturnError(400, 'Bad session');
    }
    break;
// Telegram bot
case 'mess_auth':
    if($context->method == "POST") {
        $sess_data = Telegram::CreateSession(
            $context->post['device_id'],
            $context->post['token']
        )
        return RestUtil::ReturnJson( $sess_data );
    } else {
        return RestUtil::ReturnError(400, 'Bad request');
    }
    break;
case 'mess_list_devices':
    if( Telegram::IsSessionValid(
        $context->post['token']
    ) ) {
        $result = Device::List(
```

```
                Telegram::GetUserID(),
            )
            return RestUtil::ReturnJson( $result );
        } else {
            return RestUtil::ReturnError(400, 'Bad session');
        }
        break;
    case 'mess_list_alarms':
        if( Telegram::IsSessionValid(
            $context->post['token']
        ) ) {
            $result = Device::ListAlarms(
                Telegram::GetUserID(),
                true
            )
            return RestUtil::ReturnJson( $result );
        } else {
            return RestUtil::ReturnError(400, 'Bad session');
        }
        break;
    // Default mode
    default:
        http_response_code(404);
        RestUtil::ReturnError(404, 'Not found');
        die();
}
}
$application = new RestUtil('API_Router');
$application->Render();
```

ДОДАТОК Є

Ілюстративна частина

«Одномодульна мультисенсорна система безпеки»

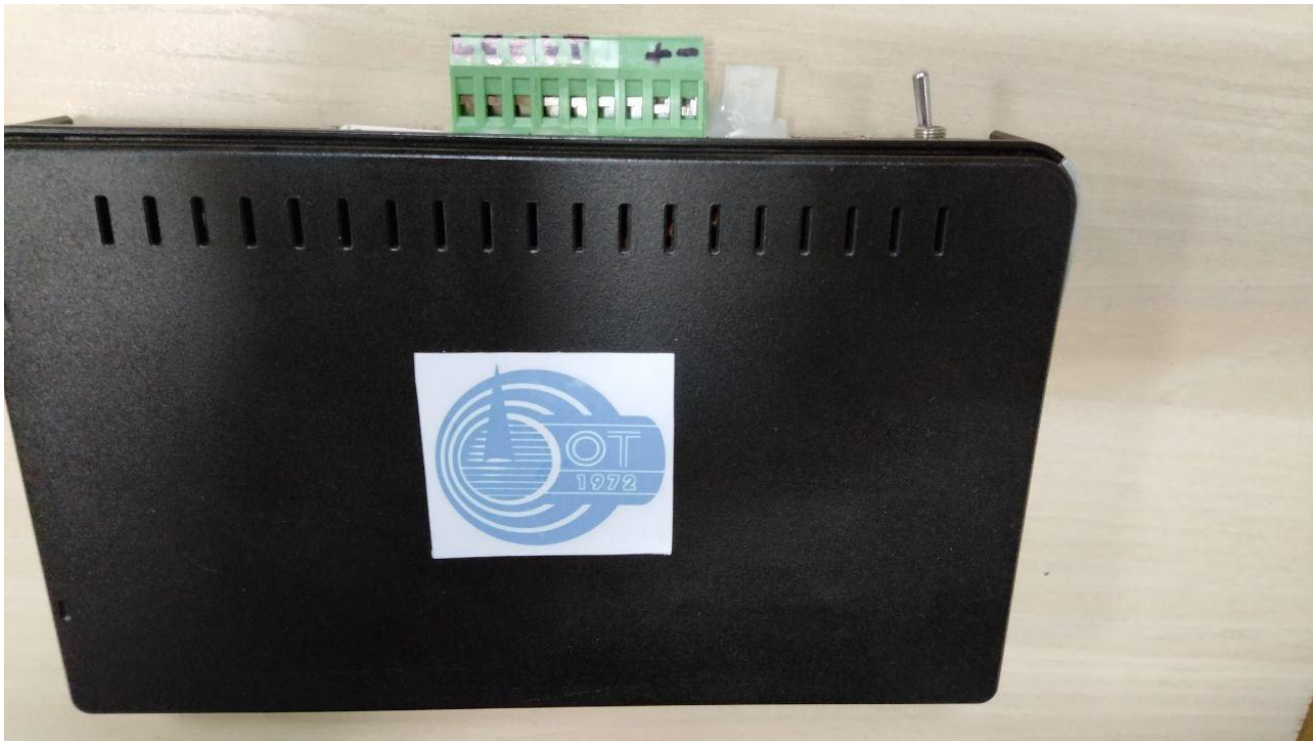
Перелік ілюстративних матеріалів:

1) Зображено пристрій системи безпеки
2) Робота GSM мережі
3) Графік залежності станів від опору шлейфу
4) Схема підключення шлейфів до Arduino NANO
5) Реалізація схеми контролю напруги
6) Схема мережевої взаємодії
7) ER-діаграма бази даних для серверного компонента
8) Налаштування контролеру через браузер

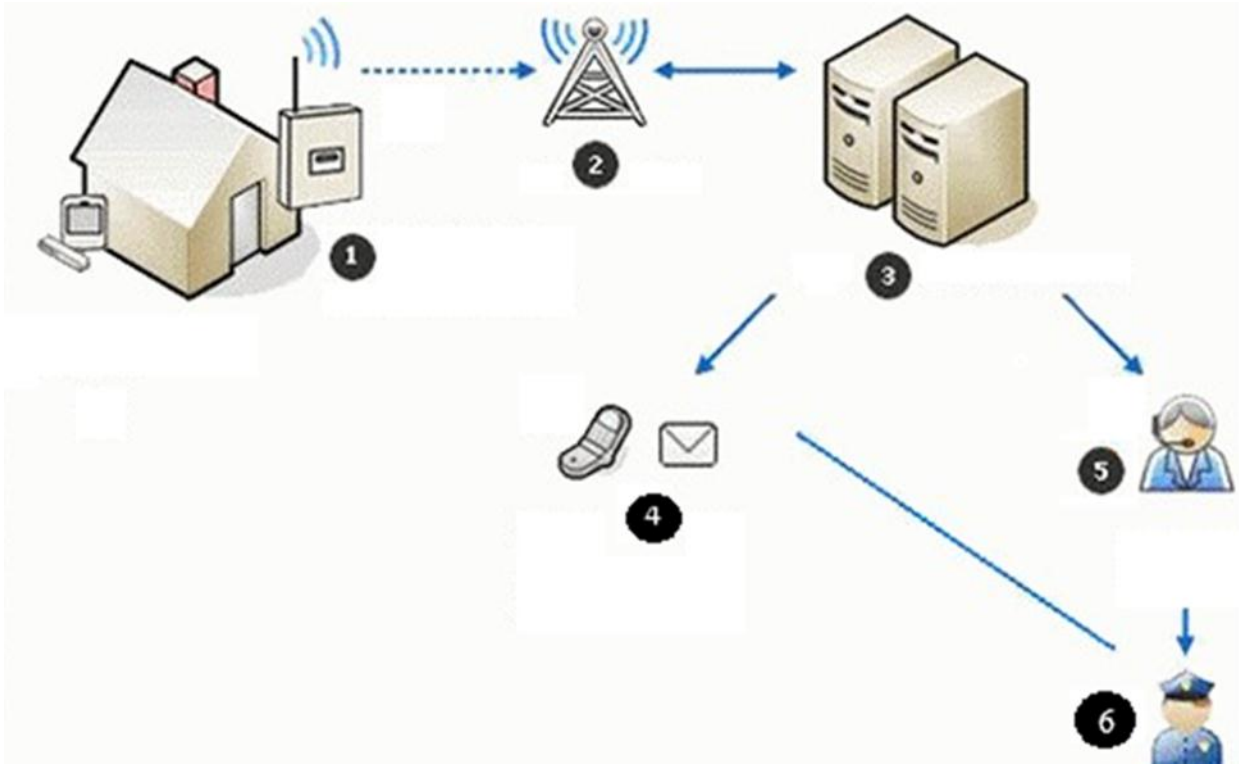
Виконав: студент 2-го курсу, групи 1КІ-20мс
спеціальності 123 — Комп’ютерна інженерія
___Ярошевський М.М
Керівник: к.т.н., доцент кафедри ОТ
_Томчук М.А
«____» _____ 2022 р.
Рецензент: к.т.н.,доцент кафедри ЗІ
_____Куперштейн Л.М
«____» _____ 2022 р.

Вінниця 2022

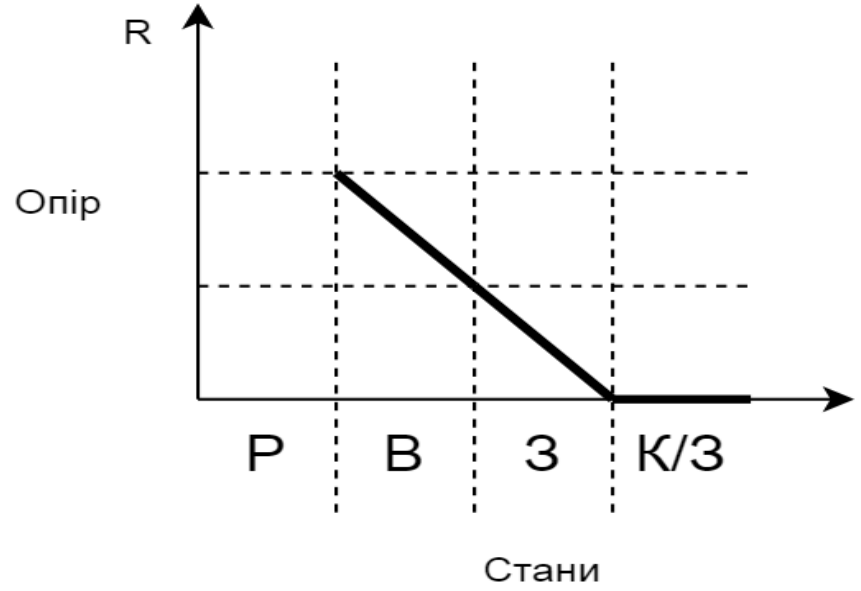
1) Зображено пристрій системи безпеки



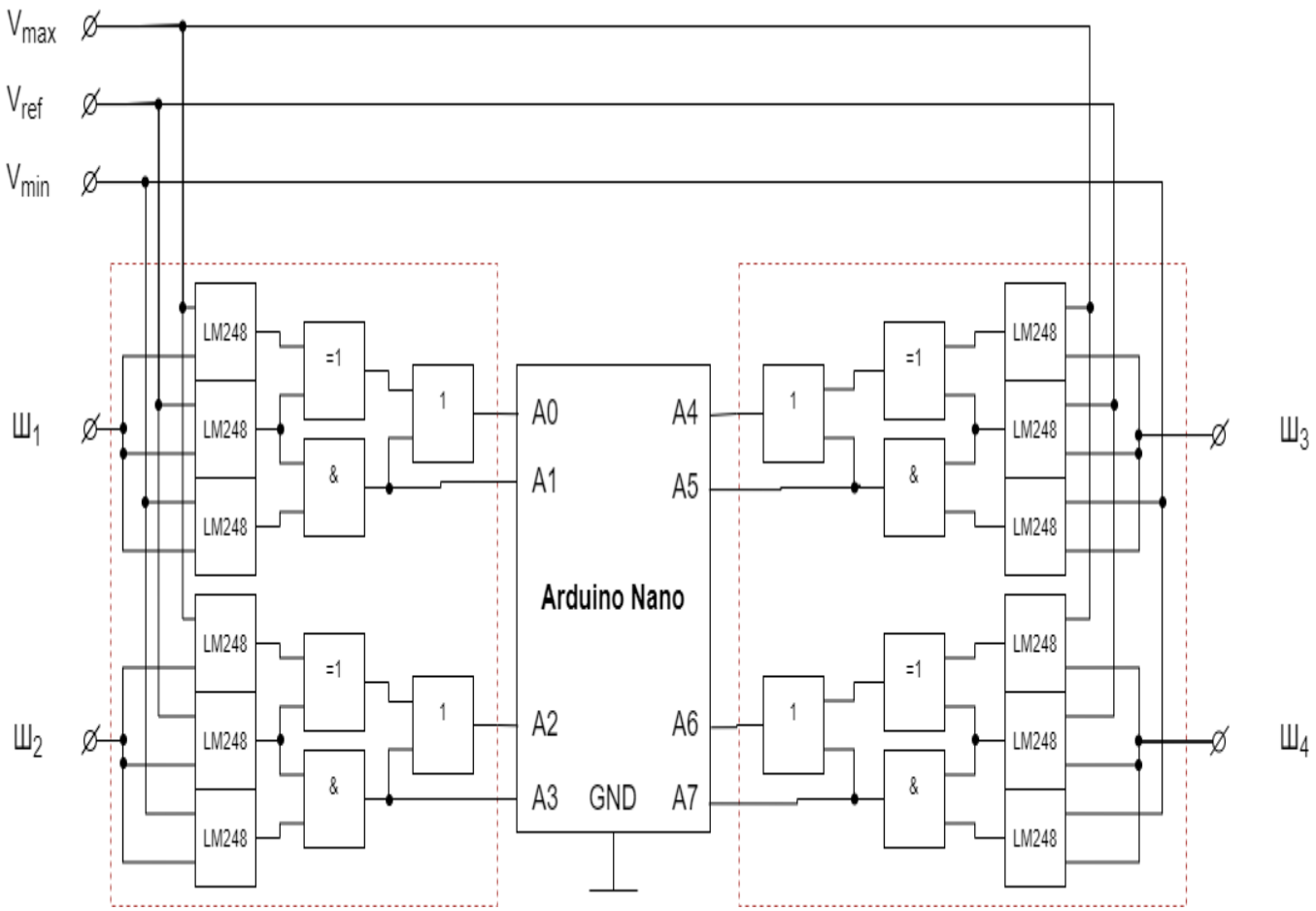
2) Робота GSM мережі



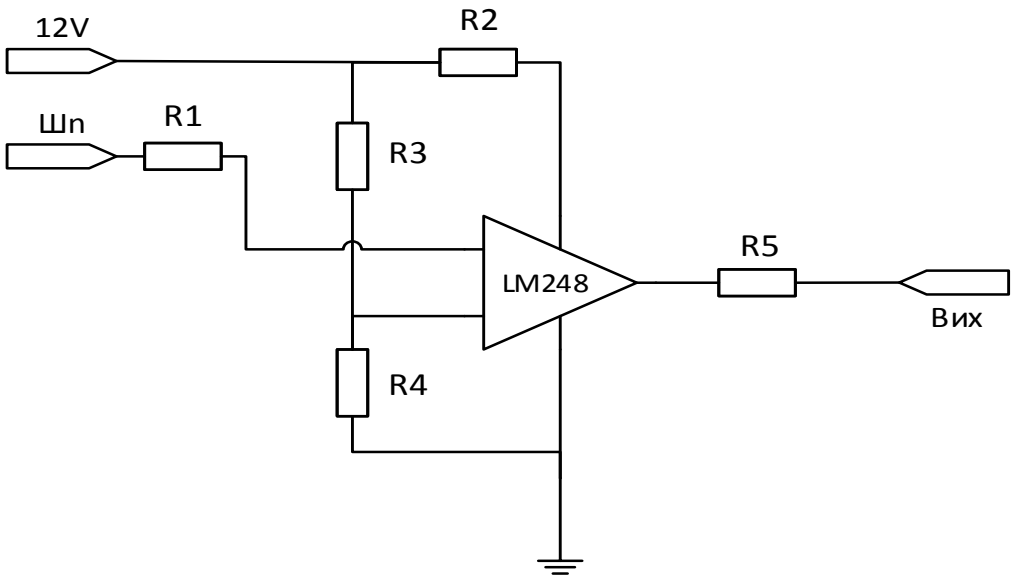
3) Графік залежності станів від опору шлейфу



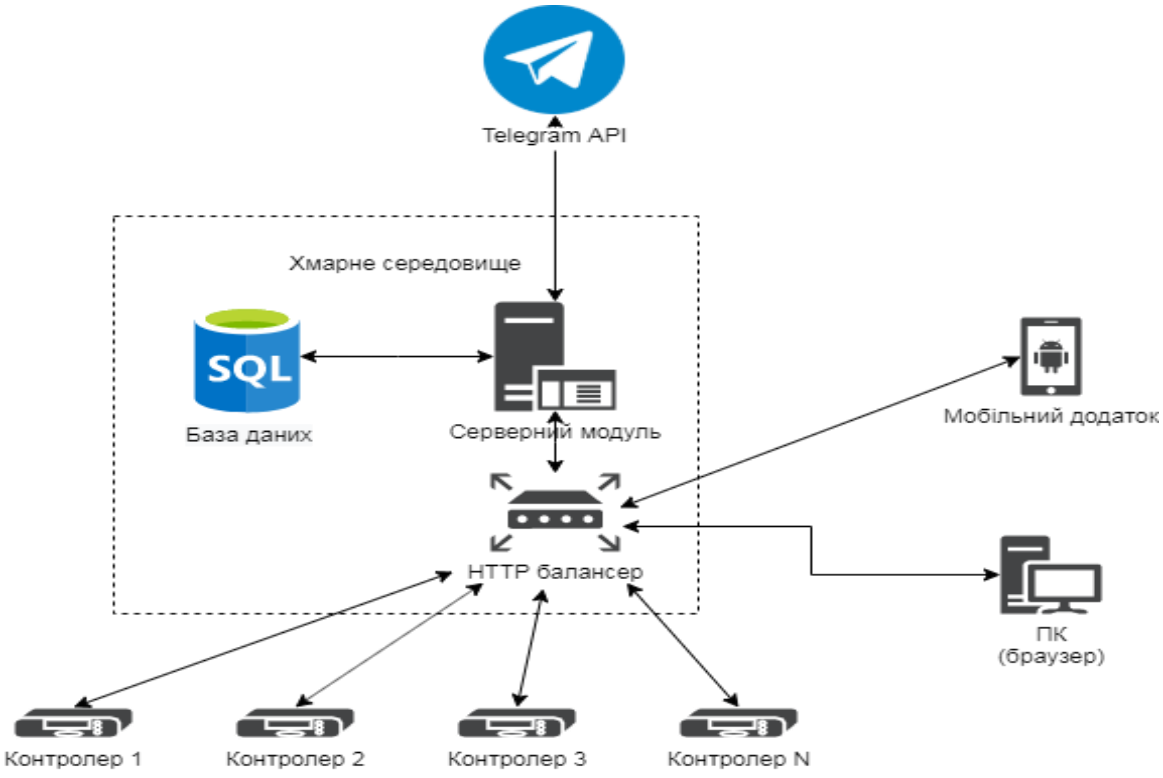
4) Схема підключення шлейфів до Arduino NANO



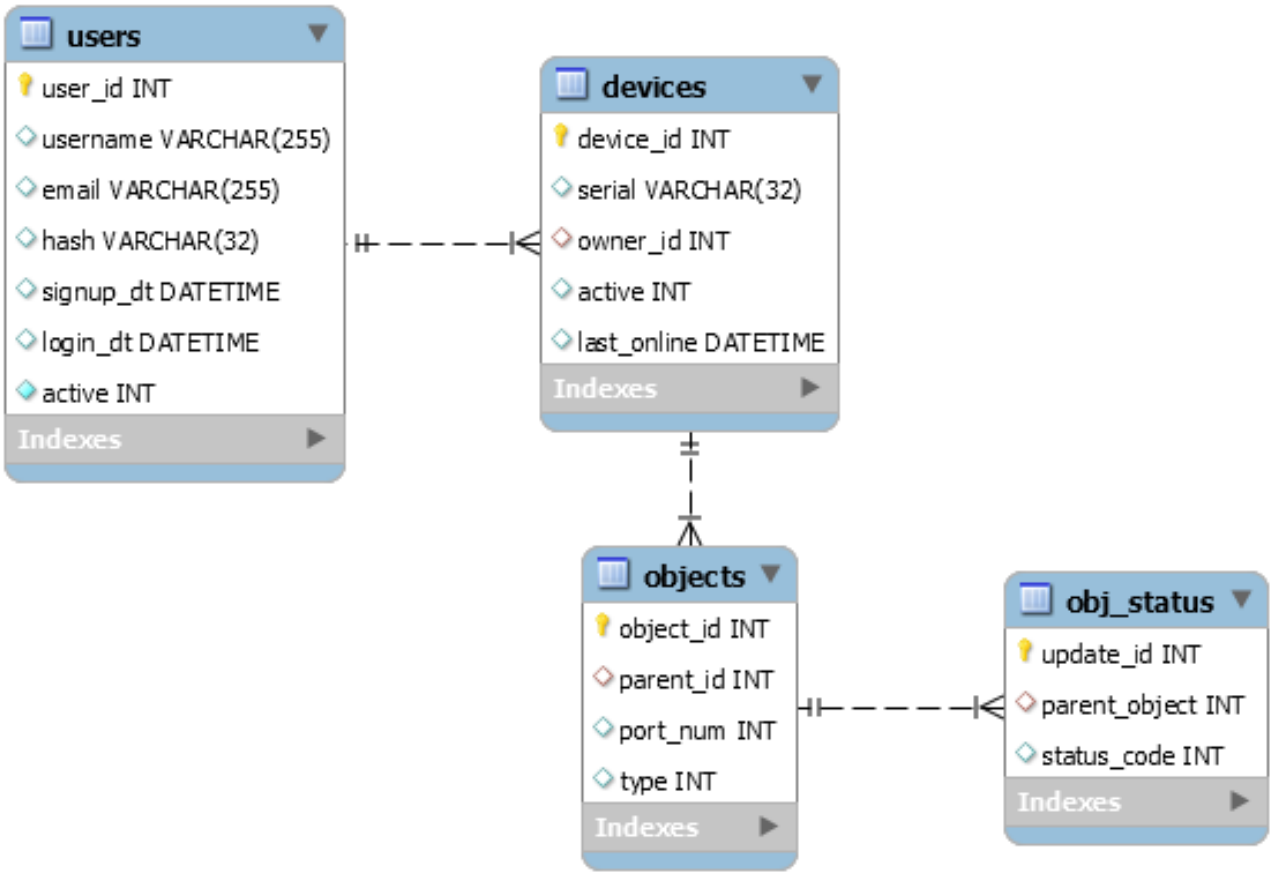
5) Реалізація схеми контролю напруги



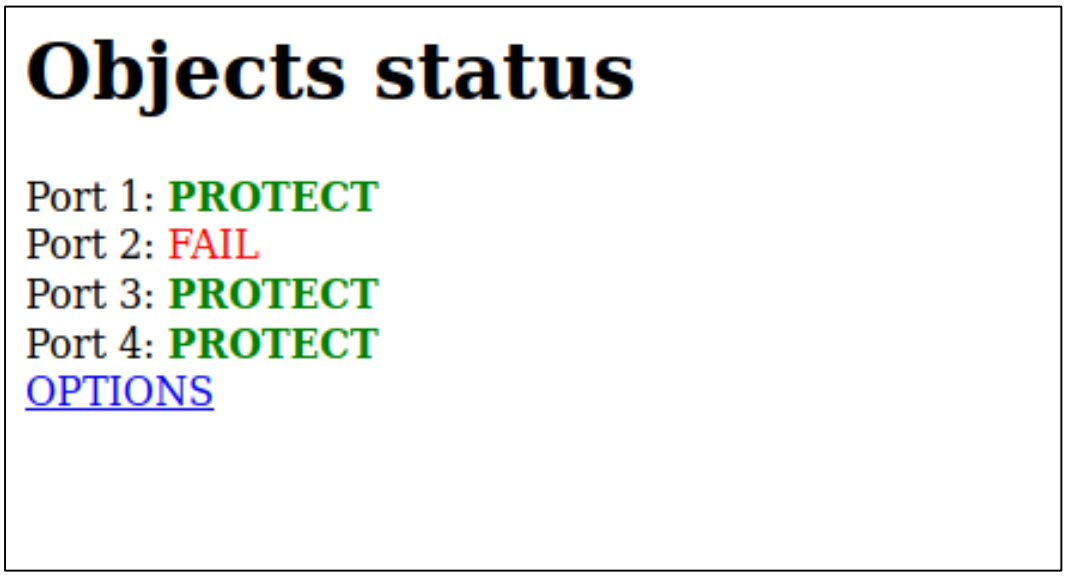
6) Схема мережевої взаємоді



7) ER-діаграма бази даних для серверного компонента



8) Налаштування контролеру через браузер



ДОДАТОК Ж

ПРОТОКОЛ

ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ
ЗАПОЗИЧЕНЬ

Назва роботи: «Одномодульна мультисенсорна система безпеки»

Тип роботи: бакалаврська дипломна робота

Підрозділ кафедра обчислювальної техніки

Показники звіту подібності Unicheck

Оригінальність 68,7% Схожість 31,3%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Ярошевський М. М.

Керівник роботи _____ Томчук М. А.