МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 20__ р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Модульна система розподіленого обміну інформації

Освітньо-кваліфікаційний рівень "бакалавр"
Спеціальність 6.050102 – "комп'ютерна інженерія"

Керівник проекту:	_____ (підпис)	Барбарук Л.В. (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Критська Я.О. (ініціали, прізвище)
Здобувач вищої освіти:	_____ (підпис)	Попов Я.В. (ініціали, прізвище)
Група:		КІ-14ад

Сєвєродонецьк 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки 6.050102 – "комп'ютерна інженерія"
(шифр і назва)
Спеціальність
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри КНІ
І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Попову Ярославу Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Модульна система розподіленого обміну інформації

керівник проекту (роботи) Барбарук Ліна Вікторівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " " 201_ р. № ____
2. Термін подання студентом роботи 16.06.2018
3. Вихідні дані до роботи матеріали переддипломної практики, література у галузі розробки додатків під Android, теоретичні відомості про інтернет речей, середа розробки Jet Brains Intellij Idea, Android Studio

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Огляд засобів розробки додатків, проектування системи контролю інформаційного оточення, практична реалізація системи, охорона праці

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	ст. викл. Критська Я.О.		

7. Дата видачі завдання 30.04.2018

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання на дипломне	05.05.2018 – 13.05.2018	
	проектування		
2	Аналіз завдання та робота з літературою.	14.05.2018 – 01.06.2018	
3	Розробка програмної системи		
4	Тестування програмної системи	01.06.2018 – 10.06.2018	
5	Розробка розділу «Охорона праці»	11.06.2018 – 13.06.2018	
6	Оформлення пояснювальної записки та	13.06.2018 – 16.06.2018	
	електронних плакатів		

Здобувач вищої освіти _____ Попов Я.В.
(підпис) (прізвище та ініціали)

Керівник _____ Барбарук Л.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка випускної кваліфікаційної роботи бакалавра: 66 с., 19 рис., джерел посилань 12.

Метою дипломної роботи бакалавра є розробка Android та Java додатків, які разом будуть відслідковувати, збирати, зберігати та відображати необхідну йому інформацію.

Створена програмна модель у середовищах Jet Brains Intellij Idea та Android Studio.

Ключові слова: додаток, клієнт, сокет, сенсор.

Умови одержання дипломного проекту: СНУ ім. В. Даля,
пр. Центральний 59-А,. м. Сєвєродонецьк, 93400.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – База Даних.

ОС – Операційна система

ПК – Персональний Комп'ютер

СКЗІО – Система Контролю Змін Інформаційного Оточення

API – Application Programming Interface

App, application – додаток

IDE – Integrated Development Environment

FTP – File Transfer Protocol

GPS – Global Positioning System

HTTP – Hyper Text Transfer Protocol

IP – Internet Protocol

JDK – Java Development Kit

JSON – JavaScript Object Notation

MIME – Multipurpose Internet Mail Extensions

SDK – Software Development Kit

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

ВСТУП

Сучасний світ являє собою нескінченно рухому і мінливу систему, з нескінченним набором складових її елементів: люди і машини, механізми та агрегати, начальники та менеджери. Кожен елемент системи, виконує поставлене йому завдання, спираючись на результат роботи попередніх йому елементів і складає інформаційний набір та визначає роботу наступних йому елементів. Будь то ГПС мітка на карті і маршрут пройдений зачепи, або ж температура верстата і кількість виготовлених виробів. Грунтуючись на цих відомостях, будуть зроблені висновки про ефективність роботи системи, будуть прийняті рішення про необхідність модернізації та її напрямках.

Але що ж робити, що б вчасно і якісно реагувати на зміни в інформаційному середовищі, отримувати потрібний потік інформації і реагувати тільки на необхідні відомості? Потрібна система, яка відстежувала б тільки інформацію, необхідну конкретному користувачеві. Ця система повинна бути проста і універсальна, що б вона могла бути застосована з абсолютно різними наборами даних, виконуючи основні функції з відстеження, збирання, зберігання, і відображення інформації, а так само повинна бути досить легковагої, для розміщення в будь-якому пристрої, від телефону до потужного ПК.

Дивлячись на сьогоднішній мобільний телефон, на яку-небудь до межі функціональну і модну модель, можна подумати, що це - межа досконалості, але згадайте, як ми дивилися, наприклад, на "цеглину" від ericsson році так в 1999-му і думали то ж саме. Напевно, дивлячись на телефон в 2020-му, ми з посмішкою будемо згадувати "досконалі" апарати 2015-го.

Основні завдання телефону минулого - здійснювати і приймати дзвінки, писати sms. Сьогодні ці завдання доповнилася роботою з інтернет-ресурсами, прослуховуванням музики, фотозйомкою, використанням ігор і додатків. Мобільний майбутнього, швидше за все, додасть до списку

основних завдань перегляд телепрограм, управління різною технікою, функції контролю стану здоров'я свого власника і можливо багато чого іншого.

Однак навіть зараз можливості мобільних телефонів, можуть дозволити нам перетворити мобільні пристрої в сенсори, які будуть непомітно для людини займатися збором необхідної інформації, передаючи її серверу, який буде зберігати її і аналізувати, після чого її візьме вихідний модуль, який відобразить зміни в інформаційному оточенні користувачеві. Можливості мови програмування Java дозволять зробити сенсором не тільки телефон, але і всілякі embedded-пристрої, будь то домашня мікрохвильовка, або ж високотехнологічний апарат на металургійному заводі.

1 ОГЛЯД ЗАСОБІВ РОЗРОБКИ ДОДАТКІВ

1.1 Інструментарій розробки

Для розробки будь-якго простого додатка або складної системи кожному програмісту необхідний набір інструментів:

- IDE (integrated development environment) – система програмних засобів, використовувана програмістами для розробки програмного забезпечення;
- SDK (software development kit) – комплект засобів розробки, який дозволяє фахівцям з програмного забезпечення створювати додатки для певного пакету програм, програмного забезпечення базових засобів розробки, апаратної платформи, комп'ютерної системи, ігрових консолей, операційних систем та інших платформ;
- а також додаткові інструменти для роботи з базами даних.

У роботі була обрана IDE Jet Brains Intellij Idea для роботи безпосередньо з Java і Android Studio для розробки Android додатка;

1.1.1 Середа разробки Intellij Idea

Перша версія IntelliJ IDEA з'явилася в січні 2001 року і швидко набула популярності, як перша Java IDE з широким набором інтегрованих інструментів для рефакторінга, які дозволяли програмістам швидко реорганізовувати вихідні тексти програм. Дизайн середовища орієнтований на продуктивність роботи програмістів, дозволяючи їм сконцентруватися на розробці функціональності, в той час як IntelliJ IDEA бере на себе виконання рутинних операцій.

Починаючи з шостої версії продукту IntelliJ IDEA надає інтегрований інструментарій для розробки графічного інтерфейсу користувача.

Серед інших можливостей, IntelliJ IDEA добре сумісна з багатьма популярними вільними інструментами розробників, такими як CVS, Subversion, Apache Ant, Maven і JUnit. У лютому 2007 року розробники IntelliJ анонсували ранню версію плагіна для підтримки програмування мовою Ruby.

Починаючи з версії 9.0, IntelliJ IDEA доступна в двох версіях: Community Edition і Ultimate Edition. Community Edition є повністю вільною версією, доступною під ліцензією Apache 2.0. У ній реалізована повна підтримка Java SE, Groovy, Scala, а також інтеграція з найбільш популярними системами управління версіями. У версії Ultimate Edition реалізована підтримка Java EE, UML-діаграм, підрахунок покриття коду, а також підтримка інших систем управління версіями, мов і фреймворків.

1.1.2 Середа розробки Android Studio

Android Studio - це інтегроване середовище розробки (IDE) для роботи з платформою Android, анонсована 16 травня 2013 на конференції Google I/O. IDE перебувала у вільному доступі починаючи з версії 0.1, опублікованій в травні 2013, а потім перейшла в стадію бета-тестування, починаючи з версії 0.8, яка була випущена в червні 2014 року. Перша стабільна версія 1.0 була випущена в грудні 2014 року. IDE заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains.

1.1.3 JDK та Android SDK

Java - об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Зараз мовою займається компанія Oracle, яка придбала Sun Microsystems у 2009 році. Синтаксис мови багато в чому схожий на C та C++. У офіційній реалізації, Java програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім, Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Java Development Kit (скорочено JDK) - безкоштовно поширюваний компанією Oracle Corporation (раніше Sun Microsystems) комплект розробника додатків мовою Java, що включає в себе компілятор Java (javac), стандартні бібліотеки класів Java, приклади, документацію, різні утиліти і виконавчу систему Java (JRE). До складу JDK не входить інтегроване середовище розробки на Java, тому розробник, що використовує тільки JDK, змушений використовувати зовнішній текстовий редактор і компілювати свої програми, використовуючи утиліти командного рядка.

За допомогою JDK буде розроблена основна частина СКЗИО, алгоритми збору та передачі даних, сервер прийому, зберігання, та видачі інформації, протокол спілкування клієнта-сервера, протокол роботи з базою даних.

Android SDK – комплект розробника Andoroid додатків, який включає у себе набір классів на мові Java, необхідних для створення Android додатка,

в нього входить емулятор нашої мобільної операційної системи, так що для програмування і налагодження навіть необов'язково мати під рукою пристрій на базі Android. За його допомогою буде створено Android додаток, котрий буде виконувати функції модуля-сенсора, який буде збирати інформацію.

1.2 Розробка Android додатка

Додатки під операційну систему Android розробляються в основному з використанням Java. Скомпільований програмний код упаковується у спеціальный файл-архів, Android Package. Цей файл має розширення * .apk і упаковується спеціальною утилітою aapt tool. Саме він надалі поширюється як програма і інсталюється на мобільні пристрої. Один такий файл пов'язаний з кодом однієї програми. І кожен додаток в Android живе у своєму власному світі - в такій машині. За замовчуванням, кожна програма виконується в своєму власному процесі, управлінням якого займається ядро Linux, яке також здійснює менеджмент пам'яті. Таким чином, найчастіше код програми виконується в ізоляції від усіх інших додатків. Android стартує процес, коли виникає необхідність виконати який-небудь програмний код і завершує його, коли в ньому більше немає необхідності і системні ресурси потрібні іншим додаткам. За замовчуванням, кожному додатку присвоюється свій унікальний ID Linux-користувача. Права доступу встановлюються таким чином, щоб файли програми були видні тільки цьому користувачеві і даному додатку. Хоча є способи, що дозволяють експортувати їх в інші програми. Наприклад, існує можливість «поділу» одного і того ж користувальницького ID між двома додатками. У такому випадку, вони зможуть бачити файли один одного. Для того, щоб економити системні ресурси, додатки з однаковим ID можна також домовитися

запускати в одному і тому ж Linux-процесі, розділяючи одну і ту ж віртуальну машину.

Діяльність (Activity) являє собою візуальний інтерфейс користувача - вікно. Зазвичай вона займає повністю весь екран мобільного пристрою. Діяльність може використовувати додаткові спливаючі вікна. Додаток може мати кілька активностей, при переході на іншу активність - попередня заморожується, а сама вона вноситься і зберігається в стек діяльностей.

Служба (Service) не має користувальницького інтерфейсу і виконується у фоновому режимі протягом невизначеного періоду часу. Служба буде виконуватися в системі до тих пір, поки не виконає свою роботу. Додатки можуть підключатися до служби і за допомогою інтерфейсу управляти нею.

Приймач широкомовних повідомлень (Broadcast Receiver) - компонент, що дозволяє приймати дані від зовнішніх подій і реагувати на них. Ініціалізувати передачі можуть інші програми і служби. Приймачі широкомовних повідомлень не мають інтерфейсу користувача. Однак вони можуть запустити діяльність у відповідь на отриману подію.

Контент-провайдер (Content Provider) надає певний набір даних доступним для програми. Дані можуть бути збережені у файловій системі, в базі даних або будь-яким іншим способом.

1.3 Засоби роботи із базами даних

База даних - це організована структура, призначена для зберігання інформації. У сучасних базах даних зберігаються не тільки дані, але й інформація. З поняттям бази даних тісно пов'язане поняття системи управління базою даних (СУЮБ). Це комплекс програмних засобів, призначених для створення структури нової бази, наповнення її вмістом, редагування вмісту і візуалізації інформації. Під візуалізацією інформації бази розуміється відбір відображуваних даних відповідно за заданим критерієм, їх упорядкування, оформлення і наступна видача на пристрої виводу або передачі по каналах зв'язку.

У світі існує безліч систем управління базами даних. Незважаючи на те, що вони можуть по-різному працювати з різними об'єктами і надають користувачу різні функції й засоби, більшість СУБД спираються на єдиний усталений комплекс основних понять. Це дає нам можливість розглянути одну систему й узагальнити її поняття, прийоми і методи на весь клас СУБД.

При розробці СКЗИО буде використано хмарне сховище даних Parse. Дане сховище даних має власний гнучкий інтерфейс для збереження і вибірки інформації в базі даних, підтримуваний декількома платформами, в тому числі Android. За допомогою Parse ми зможемо уніфікувати роботу з базою даних, не прив'язуючись до конкретної реалізації бази даних, створюючи індивідуальну таблицю для збереження інформації для кожної окремо взятої реалізації системи.

1.4 Постановка задачі

Необхідно створити таку систему яка буде складатись із трьох частин. Перша частина – модуль-сенсор, який буде виконувати збір даних та відсилати їх серверу. Цей модуль буде виконано як додаток на Android. Друга частина – сервер, яка буде приймати дані від модулей-сенсорів, які будуть розташовані на багатьох Android пристроях, збирати ці дані та зберегати їх у базі даних. Третя частина – модуль оцінки та відображення зібранної информациі, також буде виконаний як Android додаток.

2 ПРОЕКТУВАННЯ СИСТЕМИ КОНТРОЛЮ ЗМІН ІНФОРЦІЙНОГО ОТОЧЕННЯ

2.1 Типові засоби передачі інформації

Internet побудований за багаторівневим принципом, від фізичного рівня, пов'язаного з фізичними аспектами передачі двійкової інформації, і до прикладного рівня, що забезпечує інтерфейс між користувачем і мережею.

TCP (англ. Transmission Control Protocol, протокол управління передачею) - один з основних протоколів передачі даних Інтернету, призначений для управління передачею даних в мережах і підмережах TCP/IP. Виконує функції протоколу транспортного рівня в стеку протоколів IP.

Механізм TCP надає потік даних з попередньою установкою з'єднання, здійснює повторний запит даних в разі втрати даних і усуває дублювання при отриманні двох копій одного пакета, гарантуючи тим самим, на відміну від UDP, цілісність переданих даних і повідомлення відправника про результати передачі. Реалізація TCP, як правило, вбудована в ядро ОС, хоча є й реалізації TCP в контексті програми.

Коли здійснюється передача від комп'ютера до комп'ютера через Інтернет, TCP працює на верхньому рівні між двома кінцевими системами, наприклад, браузером та веб-сервером. TCP здійснює надійну передачу потоку байтів від однієї програми на деякій комп'ютері до іншої програми на іншому комп'ютері (наприклад, програми для електронної пошти, для обміну файлами). TCP контролює довжину повідомлення, швидкість обміну повідомленнями, мережевий трафік.

HTTP (HyperText Transfer Protocol, протокол передачі гіпертексту) - це протокол прикладного рівня, розроблений для обміну гіпертекстової інформацією в Internet. HTTP надає набір методів для вказівки цілей запиту,

що відправляється сервера. Ці методи засновані на дисципліні посилань, де для вказівки ресурсу, до якого повинний бути застосований даний метод, використовується універсальний ідентифікатор ресурсів (Universal Resource Identifier) вигляді місцезнаходження ресурсу (Universal Resource Locator, URL) або у вигляді його універсального імені (Universal Resource Name, URN).

Повідомлення по мережі при використанні протоколу HTTP передаються у форматі, схожому з форматом поштового повідомлення Internet (RFC-822) або з форматом повідомлень MIME (Multiperposal Internet Mail Exchange).

HTTP використовується для комунікацій між різними користувацькими програмами і програмами-шлюзами, що надають доступ до існуючих Internet-протоколам, таким як SMTP (протокол електронної пошти), NNTP (протокол передачі новин), FTP (протокол передачі файлів), Gopher і WAIS. HTTP розроблений для того, щоб дозволяти таким шлюзів через проміжні програми-сервери (proxy) передавати дані без втрат.

Протокол реалізує принцип запит / відповідь. Запитуюча програма - клієнт ініціює взаємодію з відповідає програмою - сервером і надсилає запит, який містить:

- метод доступу;
- адреса URI;
- версію протоколу;
- повідомлення (схоже за формою на MIME) з інформацією про тип переданих даних, інформацією про клієнта, що послав запит, і, можливо, з змістовною частиною (тілом) повідомлення.

Відповідь сервера містить:

- рядок стану, в яку входить версія протоколу і код повернення (успіх або помилка);

− повідомлення (у формі, схожій на MIME), до якого входить інформація сервера, метаінформація (тобто інформація про зміст повідомлення) і тіло повідомлення.

У протоколі не вказується, хто повинен відкривати і закривати з'єднання між клієнтом і сервером. На практиці з'єднання, як правило, відкриває клієнт, а сервер після відправлення відповіді ініціює його розрив.

Простий запит містить метод доступу та адреса ресурсу. Формально це можна записати так:

<Простий-Запит>: = <Метод><символ пробіл><Запитуваний-URI> <символ нового рядка>

В якості методу можуть бути вказані GET, POST, HEAD, PUT, DELETE та інші. Про найбільш поширених з них ми поговоримо трохи пізніше. В виді запитуваної URI найчастіше використовується URL-адресу ресурсу.

Приклад простого запиту: GET http://phpbook.info/

Тут GET - це метод доступу, тобто метод, який має бути застосований до запитуваного ресурсу, а http://phpbook.info/ - це URL-адресу запитуваного ресурсу.

Повний запит містить рядок стану, кілька заголовків (заголовок запиту, загальний заголовок або заголовок змісту) і, можливо, тіло запиту. Формально загальний вид повного запиту можна записати так:

<Повний запит>: = <Рядок Стану>

(<Загальний заголовок> | <Заголовок запиту> |

<Тема змісту>)

<Символ нового рядка>

[<Зміст запиту>]

Квадратні дужки тут позначають необов'язкові елементи заголовка, через вертикальну риску перераховані альтернативні варіанти. Елемент <Рядок стану> містить метод запиту і URI ресурсу (як і простий запит) і, крім того, використовувану версію протоколу HTTP. Наприклад, для

виклику зовнішньої програми можна задіяти наступний рядок стану: POST http://phpbook.info/cgi-bin/test HTTP / 1.0

У даному випадку використовується метод POST і протокол HTTP версії 1.0.

В обох формах запиту важливе місце займає URI запитуваного ресурсу. Найчастіше URI використовується у вигляді URL-адреси ресурсу. При зверненні до сервера можна застосовувати як повну форму URL, так і спрощену.

Повна форма містить тип протоколу доступу, адреса сервера ресурсу та адресу ресурсу на сервері.

У скороченій формі опускають протокол і адреса сервера, вказуючи лише місце розташування ресурсу від кореня сервера. Повну форму використовують, якщо можливе пересилання запиту іншого сервера. Якщо ж робота відбувається тільки з одним сервером, то частіше застосовують скорочену форму.

Використання http протоколу з метою організації діалогу між сервером і клієнтом можливо, однак ідеолгоческі невірно, тому що http протокол призначений для передачі гіпертексту. У системі, яка розробляється дані передаватимуться в рядковому форматі.

2.2 Загальна схема системи

Система контролю змін інформаційного оточення буде складатись із трьох взаємодіючих частин. Перша частина – модуль-сенсор, який буде розташовано на багатьох Android пристроях. Він (модуль) буде виконувати функції по збору інформації та відсилати її серверу. Друга частина – Java сервер, котрий повинен збирати інформацію, яку йому надсилають усі сенсори розташовані будь-де на будь-якому Android пристрою. Також

сервер буде зберігати дані у бази даних, для подальшого їх аналізу та відображення. Третя частина також буде виконана у виді Android-додатка та розташована на телефоні користувача системи. Цей додаток буде вибирати дані із бази даних, аналізувати та відображати користувачу у вигляді діаграм, графіків чи будь яким іншим засобом.

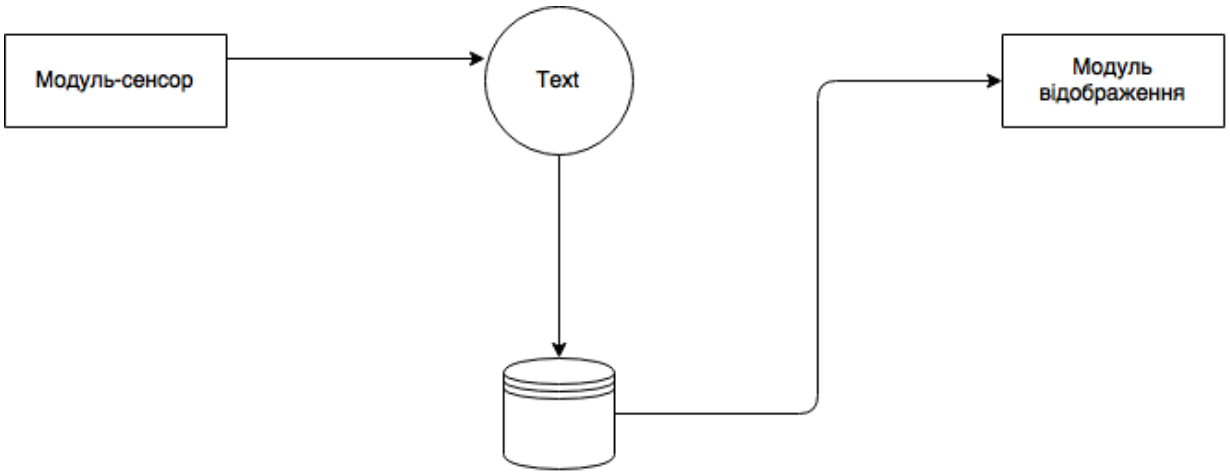


Рисунок 2.1 – Загальна архітектура СКЗИО

Як видно на рисунку 2.1 - принцип роботи системи гранично простий, що позитивно позначиться на швидкодії системи, що в свою чергу є важливим чинником, оскільки чим швидше користувач системи буде реагувати на зміни в інформаційному оточенні, тим більша ймовірність того, що користувач отримає вигоду з обставин, що склалися, і тим менше ймовірність того, що при сформованій негативної ситуації користувачу буде завдано шкоди.

Поділ системи на кілька модулів також дозволить максимально спростити завдання кожного з них. Що в свою чергу призведе до спрощення реалізації та зменшення обсягу коду кожного модуля. Що, в свою чергу, призведе до максимально можливого зниження обсягу додатку, із-за чого стане можливим його установка на застарілих пристроях, або пристроях з низьким запасом пам'яті.

2.3 Протокол спілкування клієнт-сервер

Для того щоб модулі-сенсори могли слати великі обсяги даних і для того що б ці дані не були загублені в процесі пересилання, а також не були загублені в процесі сортування, або добавленні в невірну базу, необхідно реалізувати швидкодіючий і надійний протокол спілкування модулей-сенсорів з сервером, а також реалізувати ідентифікацію сенсора і сервера. Ідентифікація необхідна для того, щоб сервер знав з якою системою йому належить спілкування, і в яку базу даних занести пакет даних, який йому буде переданий сенсором. Для цього введено ідентифікацію за двома ключами:

- ключ клієнта (модуля-сенсора);
- ключ сервера.

Ключ клієнта буде ідентифікувати всю систему в цілому і по ньому сервер буде пов'язувати всі дані, які передані йому модулями-сенсорами. Ключ сервера само необхідний для того, щоб модулі-сенсори працювали з єдиним сервером, і єдиною базою даних. Це необхідно для того щоб дані, зібрані сенсорами, не були втрачені і розіслані на різні сервера. У такому випадку модуль відповідальний за відображення інформації не зможе зібрати повний пакет зібраних даних, що в підсумку говорить про некоректну і неповну роботу системи.

Для того щоб зв'язок між сервером і модулями-сенсорами був стабільним і високошвидкісним, необхідно використовувати протокол Web Scoket. Цей протокол двостороннього зв'язку поверх TCP-з'єднання, призначений для обміну повідомленнями між клієнтом і веб-сервером в режимі реального часу. WebSocket – новий шлях для клієнтів, щоб зв'язуватися з сервером і навпаки без зайвих HTTP заголовків. WebSocket використовують свій власний протокол, який був визначений IETF. Остання версія - RFC 6455. Попередні версії протоколу довели, що він має деякі проблеми безпеки, тому він був реалізований в деяких браузерах, як Opera,

але не був включений за замовчуванням. Нові версії протоколу, вирішили ці проблеми, і браузери працюють над його підтримкою в даний час. Крім свого протоколу, веб-сокети також мають свій API, який може бути використаний в додатках, щоб відкрити або закрити з'єднання і посилати/отримувати повідомлення. Він називається WebSockets API і визначений у специфікації W3C.

З веб-сокетами можна використовувати двосторонній повнодуплексний зв'язок між сервером і клієнтом з мінімальними накладними витратами у порівнянні зі звичайним http. Веб-сокети працюють швидше, мають більш масштабовану і більш міцну високу продуктивність додатків в реальному часі. По аналізах Kaazing корпорації, використання веб сокетов може зменшити розмір трафіку заголовків http від 500 : 1 до 1000 : 1 і зменшити затримки в мережі до 3 : 1. Це призводить до серйозного підвищення продуктивності, особливо для додатків, що вимагають швидкого оновлення в реальному часі.

Оглядаючи усе що було сказано раніше, враховуючи ідентифікацію за двома ключам, ми можемо описати протокол спілкування клієнта з сервером таким чином:

- відкриття з'єднання;
- підтвердження від сервера;
- відправка ключа клієнта;
- підтвердження від сервера;
- відправка ключа сервера;
- підтвердження від сервера;
- відправка даних;
- підтвердження від сервера;
- закриття з'єднання;
- підтвердження від сервера.

Однак можливе виникнення помилки на будь-якому кроці виконання протоколу. У такому випадку сервер відповість клієнтові не

підтвердженням, а пошле помилку, відповідного типу. Виходячи з цього, необхідно модифікувати протокол спілкування як показано на рисунку 2.2.

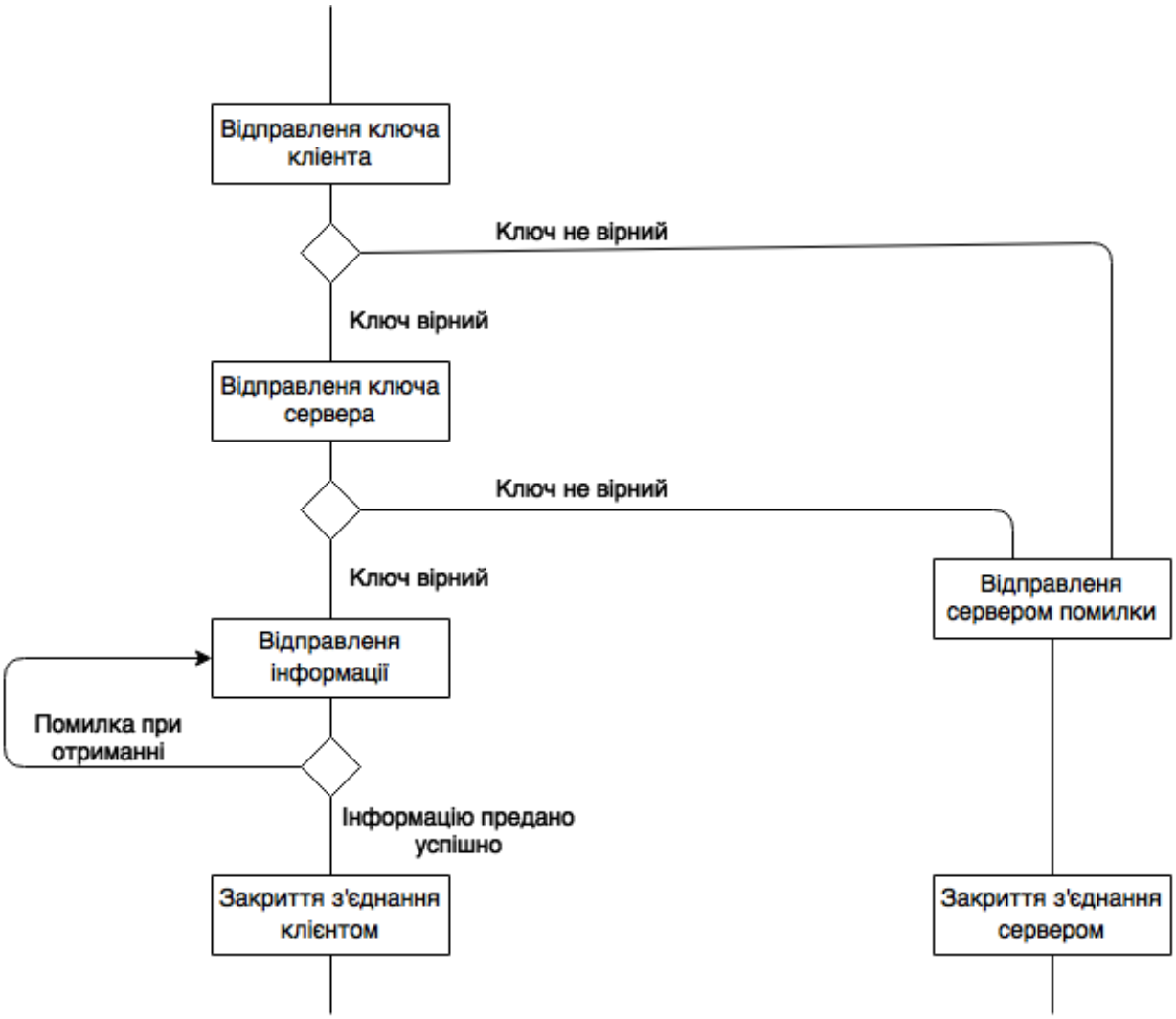


Рисунок 2.2 – Модифікована схема протоколу спілкування клієнта з сервером

Таким чином, при виникненні помилки при ідентифікації по двох ключах, сервер сам розірве з'єднання з клієнтом у примусовому порядку, попередньо надіславши клієнту відповідне повідомлення з помилкою про неспівпадіння ключа. Також якщо виникне помилка при отриманні даних від клієнта - розрив з'єднання, пошкодження пакету, сервер пошле клієнту відповідне повідомлення з помилкою, обробивши яку клієнт повторить відсилання даних на сервер. При успішному отриманні даних сервер відправить клієнту повідомлення з підтвердженням про отримання

інформації, після чого клієнт може закрити з'єднання, або повторити процедуру відправки інформації з новими даними.

2.4 Структура повідомлень

Важливим аспектом розробки також є визначення універсальної структури повідомлень, які будуть використовуватися в діалозі клієнта-сервера. Для зручності організації структури повідомлення, використовуватимемо формат JSON.

JSON (JavaScript Object Notation) – легковаговий формат обміну даними. Його з легкістю читають і пишуть люди. Також він простий для розуміння і генерації комп'ютерами. Він заснований на подветви мови програмування яваскрипт, описаний стандартом ECTA-262 3rd Edition - December 1999. JSON є повністю мовонезалежним текстовим форматом, але використовує угоди, які знайомі програмістам мов сімейства C, включаючи C, C ++, C #, Java, JavaScript, Perl, Python, і багато інших. Такі властивості роблять JSON ідеальним способом для обміну інформацією.

JSON будується на двох структурах:

- колекція пар ім'я/значення. У різних мовах, це реалізовано як об'єкт, запис, структури, словник, хеш-таблиці, шпонковим списку, або асоціативний масив;
- впорядкований список значень. У більшості мов це реалізовано як масив, вектор, список або послідовність.

Це універсальні структури даних. Практично всі сучасні мови програмування підтримують їх у тій чи іншій формі. Логично припустити, що формат даних, який є мовонезалежним, також базується на цих структурах.

У JSON вони приймають такі форми:

– Об'єкт – невпорядкований набір пар ім'я/значення. Об'єкт починається з {(лівої фігурної дужки) і закінчується} (права дужка). Кожне ім'я супроводжується: (двокрапкою) і пари ім'я/значення поділяються;

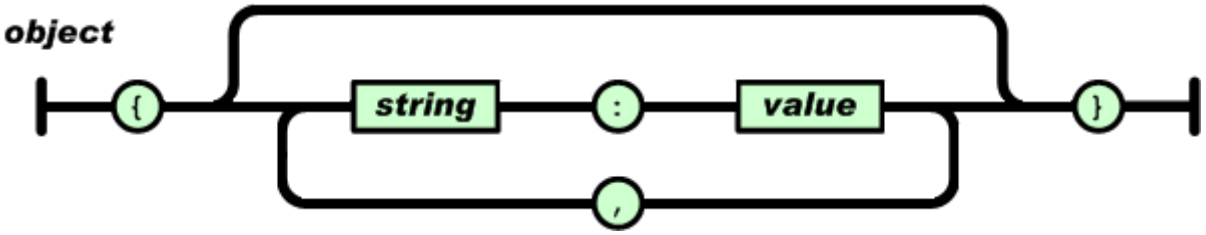


Рисунок 2.3 - JSON об'єкт

– Масив - упорядкована колекція значень. Масив починається з [(відкриває квадратної дужки) і закінчується] (закриває квадратної дужкою). Значення розділені комою.

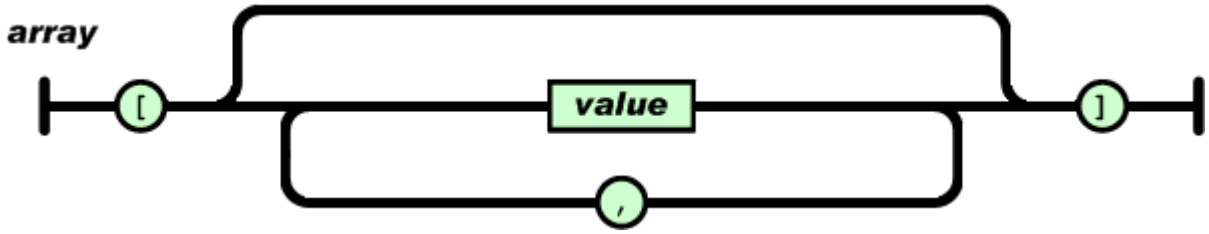


Рисунок 2.4 - JSON массив

Як показано на рисунку 2.5, значення може бути рядком у подвійних лапках, числом, true, false, null, об'єктом або масивом. Ці структури можуть бути вкладеними. Рядок - колекція нуля або більше символів Unicode, укладена в подвійні лапки, використовуючи \ (зворотну косу риску) в якості символу екранування. Символ представляється як односимвольний рядок. Це відображено на рисунку 2.6. Схожий синтаксис використовується в C і Java. Як показано на рисунку 2.7 – число представляється так само, як в C або Java, крім того, що використовується тільки десяткова система числення.

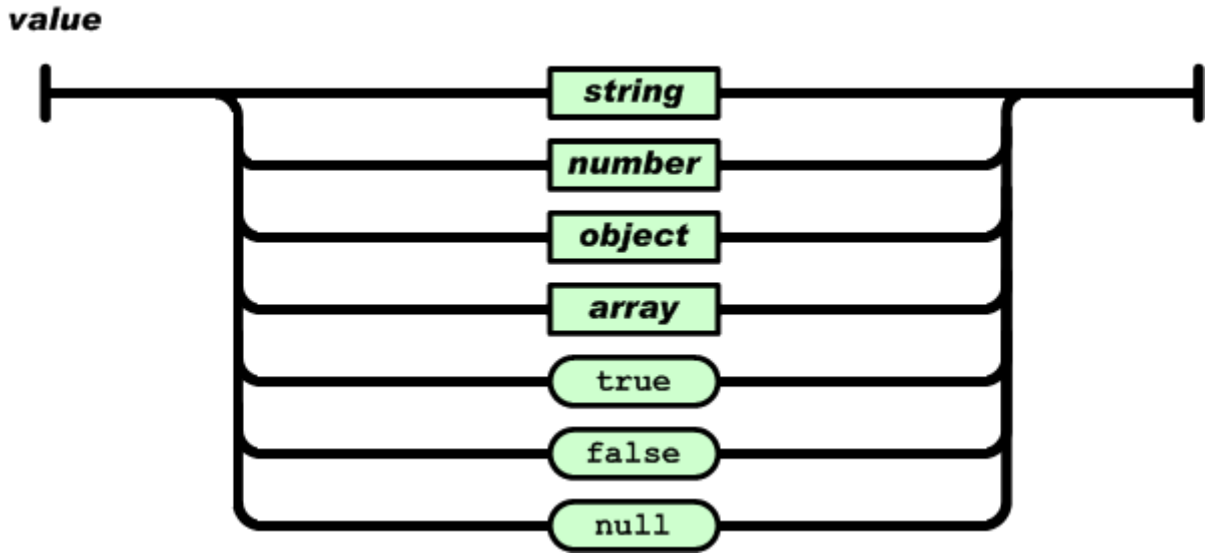


Рисунок 2.5 – Типи значень JSON

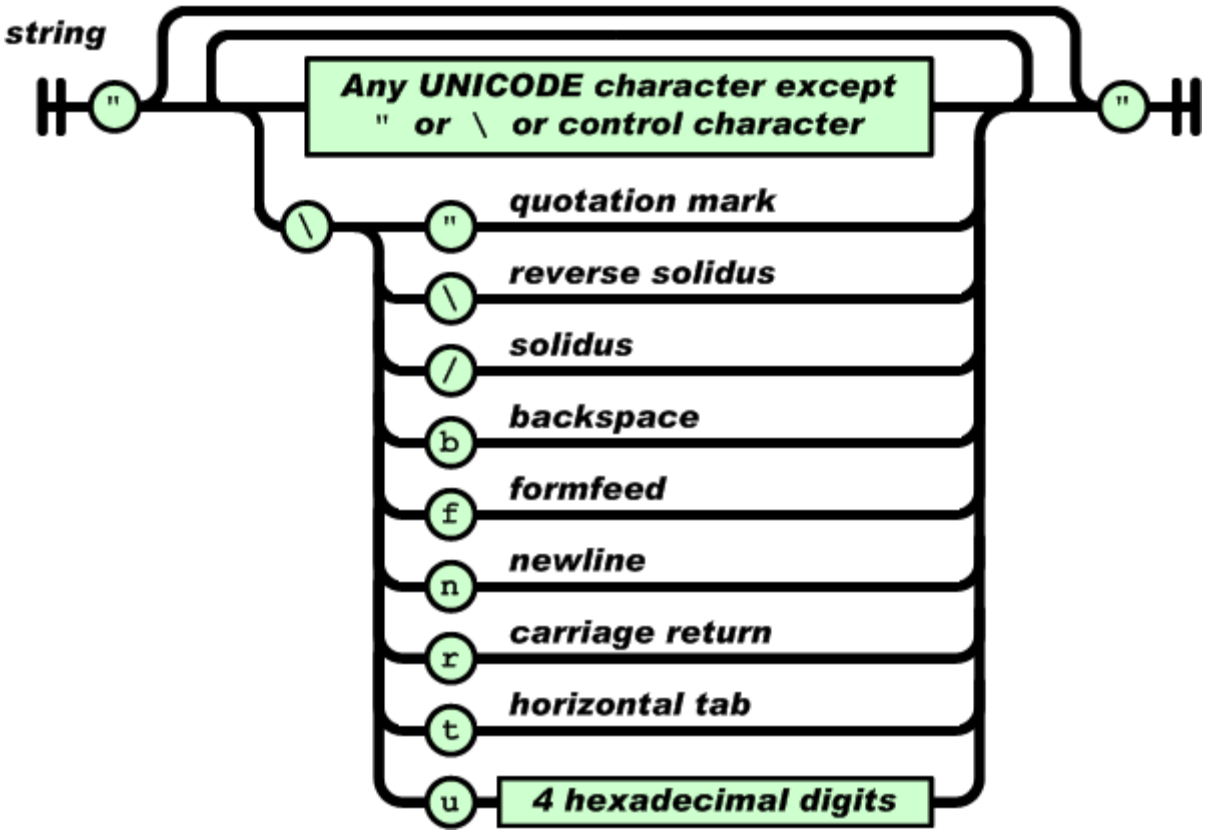


Рисунок 2.6 – Структура рядка JSON

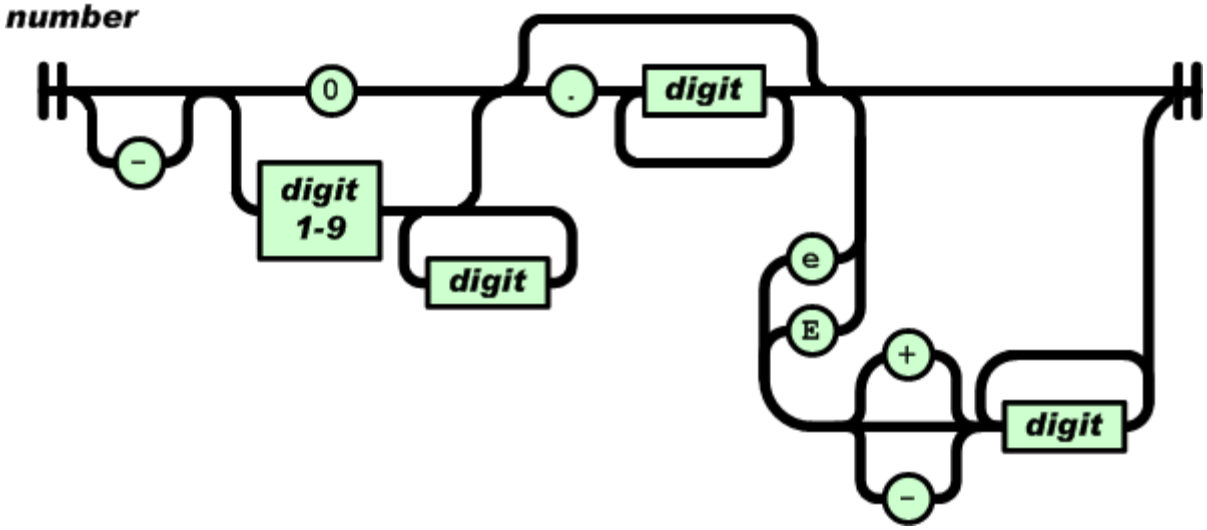


Рисунок 2.7 – Структура числа JSON.

Таким чином, використовуючи JSON як формат, визначимо універсальну структуру повідомлень:

{ "action" : <тип дій>,
"data" : <дані> }

Поле action буде указувати на те, яку інформаційне навантаження несе поточне повідомлення. Це може бути передача ключів від клієнта сервера, або повідомлення про помилку від сервера клієнту, або передача зібраної інформації від клієнта сервера.

Поле data буде містити дані, які необхідно передати при поточному параметрі action. Наприклад якщо це ключ клієнта, то повідомлення буде виглядати таким чином:

```
{ "action" : "GIVEN_CLIENT_KEY",
"data" : "SOME_GENERATED_CLIENT_KEY" }
```

Рисунок 2.8 – Приклад повідомлення від клієнта до серверу

Отримавши таке повідомлення, сервер зрозуміє за значенням параметра action що йому передають ключ клієнта, прочитає значення параметра data і порівняє його з збереженим в налаштуваннях ключем клієнта. Якщо ключ буде вірним, сервер відповість клієнту таким повідомленням:

```
{       "action" : "SERVER_RESPONSE_OK",
        "data" : ""    }
```

Рисунок 2.9 – Приклад позитивної відповіді сервера

У разі не збіга ключів, сервер відповість клієнту сполученням з описом помилки, після чого розірве з'єднання.

```
{       "action" : "ERROR_CLIENT_KEY",
        "data" : ""    }
```

Рисунок 2.10 – Приклад негативної відповіді сервера

Також слід обумовити формат упаковки інформації. Так як система позиціонується як універсальна, і сервер може приймати будь-які дані, він не буде знати, яку саме інформацію йому передадуть сенсори. Для цього, при упаковці інформації, слід упаковувати кожне передане значення в JSON-об'єкт такої струтури:

```
{       "property_name" : "<ім'я_елементу>",
        "property_type" : "<тип_елементу>",
        "property_value" : "<значення_елементу>" }
```

Рисунок 2.11 – Структура елементу даних

Після чого всі елементи такого виду будуть упаковані в JSON-масив:

```
[{      "property_name" : "<ім'я_елементу>",
        "property_type" : "<тип_елементу>",
        "property_value" : "<значення_елементу>" },
        ...,
{       "property_name" : "<ім'я_елементу>",
        "property_type" : "<тип_елементу>",
        "property_value" : "<значення_елементу>" }]
```

Рисунок 2.12 – Загальний вид упакованної інформації

Таким чином, вперше отримавши певний набір даних, сервер зможе переглянути назви елементів масив і динамічно створити таблицю в базі даних для зберігання інформації, що запобігає необхідності попереднього створення таблиці програмістом.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Реалізація модуля-сенсору

Для створення реалізації модуля-сенсора потрібно визначити його функції. У першу чергу це збір інформації, після чого зібрана інформація повинна бути відправлена на сервер. Спілкування клієнта з сервером буде виконуватись через мережу інтернет, використовуючи, як вже було визначено internet-протокол передачі інформації WebSocket, і його API. Для реалізації функції відправки на сервер, необхідно реалізувати власний протокол спілкування з сервером, заснований на інтернет-протоколі WebSocket. Що б спростити завдання реалізації протоколу спілкування клієнта з сервером, користуються open-source реалізацією інтерфейсфа WebSocket API – бібліотекою TooTallNate. Дана бібліотека потребує мінімальних витрат для реалізації множинного підключення клієнтів до сервера, спростує роботу сервера з клієнтами.

В виді необхідної інформації, будемо використовувати GPS мітку, що вказує про місцезнаходження пристрою-клієнта. Дані мітки будуть збиратися при переміщенні пристрою через кожні 50 метрів, або через кожні 5 хвилин, якщо пристрій не рухається. В результаті збору такої інформації, можна буде проаналізувати популярні маршрути, середньоденну тривалість шляху, швидкість переміщення і багато іншого.

3.1.1 Реалізація протоколу спілкування клієнта з сервером

Для опису протоколу спілкування клієнта з сервером використовуючи бібліотеку TooTallNate необхідно створити клас, який буде розширювати

клас WebSocketClient, який знаходиться в бібліотеці. Оскільки розширюваний клас - абстрактний, необхідно описати реалізацію ключових методів, а саме:

- onOpen;
- onMessage;
- onError;
- onClose.

Дані методи не використовуються програмістом безпосередньо, але використовуються системою (програмою) для передачі програмісту необхідних значень і параметрів.

- метод onOpen викличеться в той момент, коли буде успішно встановлено з'єднання двох пристроїв на основі протоколу WebSocket. Після спрацьовування даного методу можлива відправка повідомлень на сервер;
- метод onMessage буде викликатися кожен раз, коли буде надіслано повідомлення даного клієнта. Даний метод є ключовим для реалізації протоколу спілкування клієнта з сервером, оскільки коли сервер буде посилати сенсору позитивні чи негативні відповіді, саме через цей метод ми зможемо отримати їх, обробити і обрати, що робити далі;
- метод onError буде викликатися у разі непередбачених програмістом програмних помилок на стороні клієнта або сервера, викликаних зовнішніми факторами або помилкою програміста. У цьому методі слід реалізувати обробку програмних помилок, щоб уникнути некоректних завершень роботи сенсора;
- метод onClose буде викликати при розриві з'єднання сервером або клієнтом. Після спрацьовування цього методу неможливо відправлення повідомлень на сервер. Для цього необхідно буде заново відкрити з'єднання з сервером, і пройти процедуру ідентифікацію ключів клієнта і сервера.

```
@Override
public void onOpen(ServerHandshake serverHandshake) {
        System.out.print("onOpen\n");
}
@Override
public void onMessage(String s) {
        System.out.print("onMessage\n");
        if(s.equals(Action.OK_CONN.name())) {
                send(mPropertyHolder.getApiKey());
        } else if(s.equals(Action.OK_API_KEY.name())) {
        send(mPropertyHolder.getApiSecret());
        } else if(s.equals(Action.OK_API_SECRET.name())) {
        System.out.print("conn established, keys checked.");
        }else if(s.equals(Action.ERROR_API_KEY.name())) {
         onUserError(Action.ERROR_API_KEY.name());
        }else if(s.equals(Action.ERROR_API_SECRET.name())) {
        onUserError(Action.ERROR_API_SECRET.name());
        }else if(s.equals(Action.ERROR_DATA.name())) {
                onUserError(Action.ERROR_DATA.name());
        }
}
@Override
public void onClose(int i, String s, boolean b) {
        System.out.print("onClose\n");
}
@Override
public void onError(Exception e) {
        System.out.print("onError\n");
        ErrorHandler.get().handleSystemError(e);
}
```

Рисунок 3.1 – Програмна реалізація абстрактних методів WebSocket

На рисунку 3.1 наведено програмну реалізацію цих методів. На ньому видно, що в методі onMessage реалізована обробка позитивних повідомлень від сервера - установка з'єднання, підтвердження ідентифікації ключів клієнта і сервера, а також негативних повідомлень, зміст інформацію про помилки. У такому випадку викликається метод handleUserError в класі ErrorHandler, який повідомить користувача про проблему в ході роботи програми.

Теж саме відбувається в методі onError. Метод handleSystemError перехопить системний виняток, запобігши некоректне завершення програми. Після чого він опрацює його, і виведе повідомлення про помилку, що виникла в ході роботи програми користувачеві.

3.1.2 Реалізація збору інформації

Збір інформації – в даному випадку геометок, буде здійснюватись за допомогою класу LocationManager, що входить до складу Андроїд SDK. Даний клас дозволяє отримати геометкі за допомогою GPS, або за допомогою Wifi, або мережі мобільного інтернету.

На рисунку 3.2 показано, що було створено екземпляр класу LocationManager і виконано метод getLastKnownLocation, в результаті чого був отриманий екземпляр класу Location, який містить широту і довготу.

```
final LocationManager locationManager = (LocationManager)
ctx.getSystemService(Context.LOCATION_SERVICE);
long minTime = System.currentTimeMillis() - minInterval;
Location bestResult = null;
float bestAccuracy = Float.MAX_VALUE;
long bestTime = Long.MIN_VALUE;
List<String> matchingProviders = locationManager.getAllProviders();

for (String provider : matchingProviders) {
        Location location = locationManager.
getLastKnownLocation(provider);
        if (location != null) {
                float accuracy = location.getAccuracy();
                long time = location.getTime();
                if ((time > minTime && accuracy < bestAccuracy)) {
                        bestResult = location;
                        bestAccuracy = accuracy;
                        bestTime = time;          }
                else if (time < minTime && bestAccuracy ==      Float.MAX_VALUE && time >
 bestTime) {
                        bestResult = location;
                        bestTime = time;
                }}}
```

Рисунок 3.2 – Отримання геометки

3.1.3 Реалізація відправки інформації серверу

Як вказано раніше, відправка інформації на сервер відбуватиметься у форматі JSON. Для цього необхідно отриману геометкі упакувати в JSON-об'єкт, після чого дані можуть бути відправлені на сервер. Для цього необхідно скористатися методом send у класі, яким було проведено розширення класу WebSocketClient. Це показано на рисунку 3.3.

```
JSONObject message = new JSONObject();
try {
        message.put("action", Action.GIVEN_DATA.name());
        JSONObject latitude = new JSONObject();
        latitude.put("property_name", "latitude");
        latitude.put("property_value", curLocation.getLatitude());

        JSONObject longtitude = new JSONObject();
        latitude.put("property_name", "longtitude");
        latitude.put("property_value", curLocation.getLongitude());

        data.put(latitude);
        data.put(longtitude);
} catch (JSONException e) {
        e.printStackTrace();
}
client.send(message);
```

Рисунок 3.3 – Упаковка та відправлення даних

3.2 Реалізація серверу

При реалізації сервера необхідно визначити його завдання. Сервер буде:

- отримувати підключення від клієнтів;
- ідентифікувати клієнтів;
- отримувати пакети даних;
- зберігати інформаційні пакети в базі даних.

Для виконання перших трьох пунктів необхідно реалізувати протокол спілкування сервера з клієнтом. Це буде зроблено аналогічним чином з клієнтом, однак, спосіб реалізації методу onMessage буде іншим.

Для збереження даних у базу буде використовуватися Parse API, яке крім збереження вибірки і модифікації даних, дозволяє динамічно створювати таблиці. Для цього всього лише необхідно зробити збереження даних в базу, в конкретну таблицю, і якщо такої таблиці не існує - Parse сам створить її, і заповнить переданими даними. У наслідку, при збереженні даних в цю ж таблицю, відбуватиметься тільки збереження інформації, без нової ініціалізації таблиці.

3.2.1 Реалізація діалогу спілкування сервера з клієнтом

Реалізація протоколу спілкування сервера з клієнтом відрізняється від реалізації даного протоколу у клієнта лише методом onMessage. Він приведений на рисунку 3.4. На рисунку також видно, що сервер перевіряє ключі отримані від клієнта на збіг з ключами, збереженими в конфігураційному файлі.

Також в на рисунку 3.4 показано, що при отриманні пакету даних, викликається метод onData, в якому відбувається фільтрація отриманого пакету і заповнення моделі даних для збереження в базу даних.

```
@Override
public void onMessage(WebSocket webSocket, String s) {
   JSONObject jsonObject = new JSONObject(s);
   String action = jsonObject.optString("action");
   System.out.print("\naction: " + action);
   if(action.equals(Action.GIVEN_API_KEY.name())) {
      String apiKey = jsonObject.optString("data");
      System.out.print("\napi_ket: " + apiKey);
      checkApiKey(webSocket, apiKey);
   } else if(action.equals(Action.GIVEN_API__SECRET.name())) {
      String apiSecret = jsonObject.optString("data");
      System.out.print("\napi_secret: " + apiSecret);
      if(apiSecret.equals(mPropertyHolder.getApiSecret())) {
         webSocket.send(Action.OK_API_SECRET.name());
      } else {
         webSocket.send(Action.ERROR_API_SECRET.name());
      }

   } else if(action.equals(Action.GIVEN_DATA.name())) {

      JSONArray data = jsonObject.getJSONArray("data");
      if(data != null) {
         webSocket.send(Action.OK_DATA.name());
         onData(jsonObject);
      } else {
         webSocket.send(Action.ERROR_DATA.name());
      }
   }
}
```

Рисунок 3.4 – Реалізація методу onMessage на сервері

3.2.2 Збереження пакета інформації до бази даних

Відштовхуючись від зазначеної у пункті 2.4 структури повідомлень, необхідно реалізувати фільтрацію пакета даних і зберегти їх в базу даних.

Виклик методу saveToDataBase викликає збереження переданого параметром пакета даних. Збереження інформації до бази даних виконується за допомогою Parse API, та може бути виконано навіть при відсутності internet мережі.

```
private void onData(JSONArray data) {
        ParseObject dataToSave = new ParseObject(API_KEY);
        for(int i = 0; i < data.length(); ++i) {
        try {
                JSONObject jsonObject = data.getJSONObject(i);
                String propertyName = jsonObject.optString("property_name");
                String propertyValue = jsonObject.optString("property_value");

                dataToSave.add(propertyName, propertyValue);
        } catch (JSONException e) {
                e.printStackTrace();
        }
        }
         DataBaseManager.get().saveToDataBase(dataToSave);
}
```

Лістинг 3.5 – Фільтріція пакету данних

3.3 Реалізація кінцевого модулю

Для реалізації вихідного модуля необхідно визначити, яку інформацію нам необхідно отримати. Враховуючи тип необхідних даних, можна використати та отримати безліч різних графіків і діаграм:

- відобразити маршрути сенсорів на карті;
- підрахувати середню швидкість переміщення за певний відрізок часу;
- вивести загальний пройдений шлях за день;
- підрахувати середні подолану відстань за день;
- та інше.

Щоб продемонструвати роботу вихідного модуля, відобразимо на стовпчатій діаграмі загальне подолану відстань кількох сенсорів за обмежений період часу рівний тижня. Для цього зробимо вибірку з бази використовуючи Parse API і побудуємо діаграму за допомогою open source бібліотеки graphview.

На рис 3.1 показана статистика переміщення чотирьох сенсорів протягом семи днів. Найбільша відстань було пройдено в перший і останній дні - 195 кілометрів. Найменша відстань було подолано на п'ятий день четвертим сенсором - 11 кілометрів.

Також можна порахувати середню дистанцію, пройдену кожним із сенсорів за неділю.

- червоний: 103,14 километрів;
- синій: 101,85 километрів;
- зелений: 77,14 километрів;
- пурпурний: 65,85 километрів
- жовтий: 138 километрів.

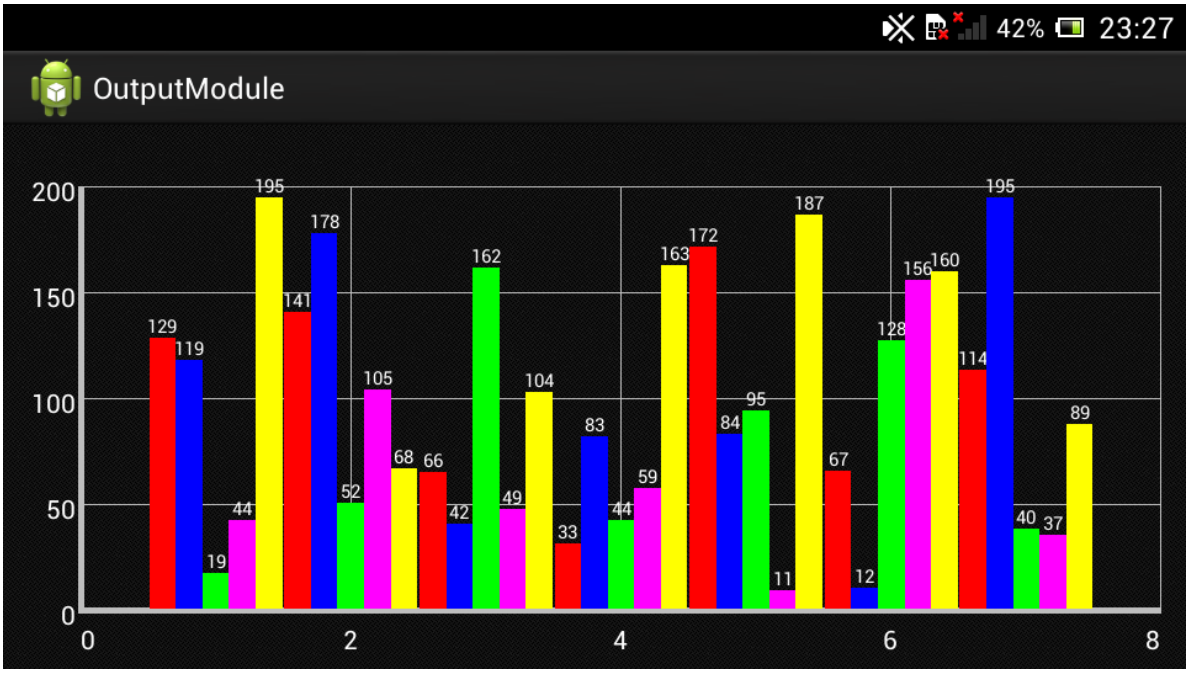


Рисунок 3.1 – Демонстрація роботи вихідного модуля

4 ОХОРОНА ПРАЦІ

4.1 Аналіз потенційно небезпечних і шкідливих виробничих факторів , що впливають на персонал

Персональні ЕОМ типу IBM PC AT має наступні характеристики:

- споживана потужність 350 Вт;
- робоча напруга 220 В;
- напруга джерел живлення +12 В, -12 В, 5 В;
- робоча частота 50 Гц.

Виходячи з приведених характеристик, очевидно, що для користувача існує небезпека поразки електричним струмом у разі недбалого поводження з комп'ютером і порушення правил експлуатації (невиконання огляду відкритих частин ПЕВМ, що знаходяться під напругою або знятих для ремонту вузлів і т. д.).

Джерелами підвищеної небезпеки можуть служити наступні елементи:

- розподільний щит;
- джерела живлення;

У відповідності з [6] до легкої фізичної роботи відносяться всі види діяльності, вироблювані сидячи і не вимагаючи фізичної напруги. Робота користувача розробленого пакету програм відноситься до категорії Іа.

Згідно з [12] приміщення для ПЕОМ по ступеню небезпеки поразки людини електричним струмом відноситься до приміщень без підвищеної небезпеки (немає струмопровідної полови, вогкості, підвищеної температури, можливості одночасного дотику до корпусів устаткування з “землею” і до струмонесучих частин).

У відповідності з [7] при обслуговуванні ПЕВМ мають місце фізичні і психофізичні небезпечні, а також шкідливі виробничі чинники:

- підвищене значення напруги в електричному ланцюзі, замикання якого може відбутися через тіло людини;
- підвищений рівень статичної електрики;
- підвищений рівень електромагнітних випромінювань;
- підвищена або знижена температура повітря робочої зони;
- підвищена або знижена рухливість повітря;
- підвищена або знижена вогкість повітря;
- відсутність або недолік природного світла;
- підвищена пульсація світлового потоку;
- недостатня освітленість робочого місця;
- підвищений рівень шуму на робочому місці;
- розумове перенапруження;
- емоційні навантаження;
- монотонність праці.

Щодо до впливу на довкілля, то програмний засіб, який було розлоблено під час дипломного прокету на довкілля ніяк не впливає.

Діяльність за темою магістерської роботи в процесі її виконання впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища», Законом України «Про забезпечення санітарного та епідемічного благополуччя населення», Законом України «Про відходи», Законом України «Про охорону атмосферного повітря», Законом України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру», Водний кодекс України.

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на відалення (знешкодження), утилізацію, тощо в ІТ галузі.

Вплив на атмосферне повітря при нормальних умовах праці не оказує, бо не має в приміщенні сканерів, принтерів та інших джерел викиду забруднюючих речовин в повітря робочої зони.

В процесі діяльності виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

- батарейки та акумулятори (малі) - III клас небезпеки;
- макулатура - IV клас небезпеки;
- матеріали пакувальні, що не вміщують целюлозу - IV клас небезпеки;
- матеріали пакувальні, що вміщують п/ет, п/пр - IV клас небезпеки;
- змінні носії інформації - IV клас небезпеки.

Наводяться вимоги зберігання виявлених за своєю роботою відходів відповідно до вимог Державних санітарних правил і норм ДСанПіН 2.2.7.029.

Відходи в міру їх накопичення збирають у тару, відповідну класу небезпеки, з дотриманням правил безпеки, після чого доставляють до місця тимчасового зберігання відходів відповідно до затвердженої схеми їх розміщення Зазначені для зберігання відходів місця чи об'єкти повинні використовуватися лише для заявлених відходів.

Не допускається зберігання відходів у невстановлених схемою місцях, а також перевищення норм тимчасового зберігання відходів.

Способи тимчасового зберігання відходів визначаються видом, агрегатним станом і класом небезпеки відходів:

- Відходи III класу небезпеки зберігаються в тарі, яка забезпечує локалізацію зберігання, дозволяє виконувати вантажно-розвантажувальні і транспортні роботи і виключає поширення в ОС шкідливих речовин;
- Відходи IV класу небезпеки можуть зберігатися відкрито на промисловому майданчику у вигляді конусоподібної купи, звідки їх автонавантажувачем перевантажують у самоскид і доставляють на місце утилізації або захоронення;

- В разі тимчасового зберігання відходів у стаціонарних складах або промислових приміщеннях повинні бути забезпечені санітарно-гігієнічними етичні вимоги до повітря робочої зони згідно з ГОСТ 12.1.005.

Не допускається змішування відходів різних видів і класів небезпеки з будівельними і побутовими відходами, відходами дерев'яної, металевої, синтетичної тари, відходами текстильних матеріалів (старий спецодяг, ганчірки) і інш.

Проведення заготовки, здачі, переробки та реалізації металобрухту встановлені окремо Законом України «Про металобрухт».

Всі відходи, що утворюються в процесі діяльності/роботи, підлягають обліку.

Вимоги безпеки при поводженні з відходами:

Під час роботи з відходами (прибирання виробничих приміщень, збір і сортування, навантаження, транспортування, розвантаження та ін.) працівники та обслуговуючий персонал підприємства повинні бути забезпечені засобами індивідуального захисту та дотримуватися вимог інструкцій з охорони праці, що діють на підприємстві.

Наведено перелік деяких відходів, які передаються на утилізацію організаціям, які мають ліцензію на поводження з відходами як вторинної сировини:

- лом і кускові відходи міді, бронзи, латуні, алюмінію, свинцю;
- брухт чорних металів;
- макулатура;
- склобій;
- матеріали текстильні вторинні;
- відходи деревини кускові;
- відпрацьовані фільтрувальні засоби індивідуального захисту;
- відпрацьовані вогнегасники;
- матеріали пакувальні вторинні.

Відвантаження таких відходів здійснюється відповідно до договору (контракту).

Побутові та будівельні відходи вивозяться на полігон твердих побутових відходів міста, також відповідно до договору з комунальним дорожньо-експлуатаційним управлінням.

Особи, винні в порушенні встановленого порядку поводження з відходами (порушення правил обліку відходів, самовільне складування і видалення відходів, передача відходів в інші підприємства/організації з порушенням встановлених правил), згідно законодавства несуть дисциплінарну, адміністративну або кримінальну відповідальність.

4.2 Заходи щодо техніки безпеки

Основним небезпечним чинником при роботі з ЕОМ є небезпека поразки людини електричним струмом, яка усугубляє тим, що органи чуття людини не можуть на відстані знайти наявності електричної напруги на устаткуванні.

Проходячи через тіло людини, електричний струм надає на нього складну дію, що є сукупністю термічної (нагрів тканин і біологічних середовищ), електролітичної (розкладання крові і плазми) і біологічної (роздратування і збудження нервових волокон і інших органів тканин організму) дій.

Ступінь ураження людини електричним струмом залежить від наступних факторів:

- значення сили струму;
- електричного опору тіла людини і тривалості протікання через нього струму;
- роду і частоти струму;

- індивідуальних властивостей людини і навколишнього середовища.

Даним проектом передбачаються наступні технічні способи і засоби, застережливі поразки людини електричним струмом:

- заземлення електроустановок;
- занулення;
- захисне відключення;
- електричне розділення сітей;
- використовування малої напруги;
- ізоляція струмоведучих частин;
- огорожа електроустановок.

Проведемо розрахунок заземлюючого пристрою.

Початкові дані для розрахунку заземлюючого пристрою:

- напруга установки, що заземляється, - 220В;
- режим нейтралу мережі - з ізольованою нейтралою;
- питомий опір ґрунту – 100 Ом·м(суглинок);
- гранично допустимий опір заземлюючого пристрою - 4 Ом;
- характеристика кліматичної зони (III):

а) середня багаторічна низька температура, оС - від –14 до -10;

б) тривалість замерзання вод, дні - 150;

в) коефіцієнт сезонності для вертикального електроду завдовжки 3м -1,5.

Визначимо розрахунковий опір ґрунту (Ом·м) по формулі (4.1).

$$\rho_{расч} = \psi \cdot \rho = 1{,}5 \cdot 100 = 150 Ом \cdot м \qquad (4.1)$$

де ρ - питомий опір ґрунту;

ψі – кліматичний коефіцієнт, що враховує стан ґрунту під час вимірювань (таблиця 4 [7]).

Розрахуємо опір розтіканню одиночного трубчастого заземлювача по формулі (4.2).

$$R_{з.1} = (\frac{\rho_{расч}}{2 \cdot \pi \cdot l}) \cdot \ln(4 \cdot \frac{l}{d}) \quad (4.2)$$

де l – довжина заземлювача (l=5м);

d – діаметр труби і стрижня (d=0,05м);

$$R_{з.1} = (\frac{\rho_{расч}}{2 \cdot \pi \cdot l}) \cdot \ln(4 \cdot \frac{l}{d}) = (\frac{150}{2 \cdot 3{,}14 \cdot 5}) \cdot \ln(4 \cdot \frac{5}{0{,}05}) = 28{,}6 \text{ Ом}$$

Розрахуємо кількість паралельно сполучених одиночних заземлювачей по формулі (4.3).

$$n = \frac{R_{з.1}}{R_{доп} \cdot \eta} = \frac{28{,}6}{4 \cdot 0.47} = 15{,}2 \quad (4.3)$$

де Rдоп=4. – самий допустимий опір заземлюючого пристрою;

η - коефіцієнт використовування ґрунтового заземлення (для шістки заземлювачей η=0,47).

Округлятимемо отримане значення у більшу сторону n=[15,2]=16.

Розрахуємо довжину горизонтальної сполучної смуги по формулі (4.4).

$$L = a \cdot (n-1) = 3 \cdot (16-1) = 45м \quad (4.4)$$

де a – відстань між вертикальними заземлювачами (a=3м);

n – кількість вертикальних заземлювачей (n=16).

Розрахуємо опір сполучної смуги по формулі (5.5).

$$R_n = \frac{\rho_{расч}}{2 \cdot \pi \cdot l} \cdot \ln(\frac{L^2}{d \cdot h}) \quad (4.5)$$

де d – еквівалентний діаметр смуги шириною l=5 (d=0,05м);

h – глибина заставляння смуги (h=0,8м).

$$R_n = \frac{\rho_{расч}}{2 \cdot \pi \cdot l} \cdot \ln(\frac{L^2}{d \cdot h}) = \frac{150}{2 \cdot 3{,}14 \cdot 5} \cdot \ln(\frac{45^2}{0{,}05 \cdot 0{,}8}) = 51{,}7 \text{ Ом}$$

Розрахуємо результуючий опір заземлюючого електроду з урахуванням сполучної смуги по формулі (4.6).

$$R_{гр} = \frac{R_{з.1} \cdot R_n}{R_{з.1} \cdot \eta_n + R_n \cdot n \cdot \eta_з} \le R_{доп} \quad (4.6)$$

де ηn – коефіцієнт використовування сполучної смуги (для 6-і заземлювачей ηn=0,27).

$$R_{гр} = \frac{R_{з.1} \cdot R_n}{R_{з.1} \cdot \eta_n + R_n \cdot n \cdot \eta_з} = \frac{26{,}6 \cdot 51{,}7}{26{,}6 \cdot 0{,}27 + 51{,}7 \cdot 16 \cdot 0{,}47} = 3{,}47 \text{ Ом}$$

3,47<4 $\Rightarrow$ умова забезпечення електробезпеки персоналу виконується.

Таким чином, остаточна кількість заземлювачей 15 шт.

4.3 Заходи, що забезпечують виробничу санітарію і гігієну праці

Підвищення працездатності людини і збереження його здоров'я забезпечується стабільними метеорологічними умовами.

Мікроклімат виробничих приміщень – це поєднання температури, вогкості і швидкості руху повітря, а також температури навколишніх поверхонь. Значне коливання параметрів мікроклімату приводить до порушення систем кровообігу, нервової і пітовидільної, що може викликати підвищення або пониження температури тіла, слабкість, запаморочення і навіть непритомність.

В приміщенні для виконання робіт операторського типу, пов'язаних з нервово-емоційною напругою, проектом передбачається дотримання наступних нормованих величин параметрів мікроклімату (див. табл. 4.1).

Таблиця 4.1 - Оптимальні параметри мікроклімату в робочій зоні виробничого приміщення для категорії робіт 1

Період року	Температура, оС	Відносна вогкість %	Швидкість руху повітря, м/с
Холодний	22.24	40.60	0,1
Теплий	23.25	40.60	0,1

Оскільки в приміщенні немає джерел виділення шкідливих речовин, можна використовувати природну вентиляцію. Площа приміщення складає 32 м2 . Для забезпечення прийнятних параметрів мікроклімату в приміщенні з такою площею можна використовувати 1 кондиціонер типу БК-2000.

Спектр випромінювання монітора комп'ютера включає рентгенівську, ультрафіолетову, інфрачервону області, а також широкий діапазон хвиль інших частот. Небезпека рентгенівського проміння нехтує мала, оскільки цей вид випромінювання поглинається речовиною екрану.

Для зниження дії електромагнітного випромінювання пропонується захист часом і відстанню. Захист часом передбачає обмеження часу перебування людини в зоні дії полів. Тривалість роботи на ПЕОМ повинна складати не більше 3.5–4.5 години.

Також необхідно забезпечити раціональне освітлення в робочому приміщенні. В проекті, що розробляється, передбачається використовувати суміщене освітлення. В світлий час доби приміщення освітлюватиметься через віконні отвори, в решту часу використовуватиметься штучне освітлення.

Штучне освітлення в робочому приміщенні передбачається здійснювати з використанням люмінесцентних джерел світла в світильниках загального освітлення, оскільки люмінесцентні лампи володіють високою світловою віддачею до 75 Лам/Вт і більш, тривалим терміном служби до 10000 годин, спектральним складом випромінюваного світла, близьким до сонячного.

Зорова робота оператора ПЕВМ відповідно до [10] відноситься до розряду Va з світловим потоком Фл=3120 кожна. Нормована освітленість на робочому місці (Ен) при загальному освітленні складає 200 лк.

Проведемо розрахунок кількості світильників в робочому приміщенні завдовжки a=6 м, шириною b=3 м, заввишки c=4 м. Формула розрахунку штучного освітлення при горизонтальній робочій поверхні методом світлового потоку (5.7):

$$Фл = \frac{Ен \cdot S \cdot Z \cdot K}{N \cdot U \cdot M} \qquad (4.7)$$

де Фл – світловий потік, Лм;

Ен – нормована освітленість;

S – площа підлоги, кв.м;

Z=1.1–1.3 - поправочний коефіцієнт світильника (для стандартних світильників);

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації світильників;

N – число світильників;

U=0.55–0.6 – коефіцієнт використовування, залежний від типу світильника, показника індексу приміщення і др.;

M – число ламп в світильнику.

З формули (4.7) виразимо N і визначимо кількість світильників для даного приміщення:

$$N = \frac{200 \cdot 18 \cdot 1{,}2 \cdot 1{,}5}{3120 \cdot 0{,}6 \cdot 2} = 1{,}7$$

Виходячи з цього, рекомендується використовувати 2 світильники. Світильники слід розміщувати рядами, бажано паралельно стіні з вікнами. Схема розташування світильників зображена на рис. (4.1).

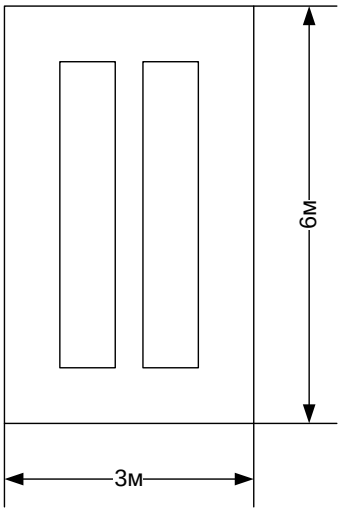


Рисунок 4.1 – Схема розташування світильників

Зниження шуму можна добитися раціонально розпланувавши приміщення, установкою устаткування на спеціальні амортизуючи прокладки. Згідно вимогам [9] рівні звуку не повинні перевищувати 50 дБ.

Для зниження стомлюваності обслуговуючого персоналу в приміщеннях, де розташовані обчислювальні засоби передбачаються використовувати спокійні колірні поєднання і покриття, що не дають відблисків. Від електромагнітного випромінювання, витікаючого від ПЕОМ, використовуються захисні екрани.

Для забезпечення чистоти повітря і відповідних мікрокліматичних умов пропонується застосувати приточування-витяжну вентиляцію. Для зменшення дії шкідливих речовин і загазованості для роботи з розплавленими матеріалами робоче місце забезпечується примусовою витяжною вентиляцією. Цей метод забезпечує притоку потрібної кількості свіжого повітря (30 мм^3 /ч на одного працюючого).

Кількість повітря, яка необхідна подавати в приміщення для забезпечення необхідних параметрів повітряного середовища, визначається на підставі кількість тепла, вологи і шкідливих речовин, що поступають в приміщення, а також враховуючи видалення повітря місцевими відсмоктуваннями від устаткування, загальнообмінною вентиляцією.

4.4 Рекомендації по пожежній профілактиці

Пожежі представляють небезпеку для життя людини і зв'язані як з матеріальними втратами, так і з відмовою засобів обчислювальної техніки, що спричиняє за собою порушення ходу технологічного процесу.

Горючими матеріалами в приміщенні, де розташовані ПЕОМ, є:

− поліамід - матеріал корпусу мікросхеми. Горюча речовина. Температура самозапалення 420 оС, енергія запалення 2мДж;

− полівінілхлорид - ізоляційний матеріал. Горюча речовина. Температура самозаймання 480 єС, енергія запалення 50мДж;

− склостоліт ДЦ - матеріал друкарської платні. Складногорючий матеріал;

− пластикат кабельний No.489 - матеріал ізоляції кабелю. Складногорючий матеріал. Температура самозаймання 1500 єС;

− плита деревостружкова - будівельний і обробний матеріал, матеріал з якого виготовлені меблі. Складнозапалений матеріал. Показник горючості 1.8;

− папір – довідкова і робоча документація, література. Горючий матеріал. Показник горючості більше 2.1.

Відповідно до [11] приміщення відноситься до категорії В (пожежовибухонебезпечної).

Джерелами запалення можуть бути:

− іскри при замиканні і розмиканні ланцюгів;

− іскри і дуги коротких замикань;

− перегріви від тривалого перевантаження і наявності перехідного опору;

− розряди статичної електрики.

Для того, щоб зупинити реакцію горіння, порушують умови її виникнення і підтримки. Звичайно для гасіння використовуються порушення двох основних умов сталого стану – пониження температури і режим руху газів. Пониження температури може бути досягнутий шляхом введення речовин, які поглинають багато тепла в результаті випаровування і дисоціації (наприклад, вода, порошки).

При повному тому, що згоряє органічних сполук утворюються С, SO, Н Про, N, а при тому, що згоряє неорганічних з'єднань – оксиди. Залежно від температури плавлення і тривалості реакції можуть знаходиться або у вигляді розплавів (Al O, Ti O), або підійматися в повітря у вигляді диму (P O, Na Про, MgO).

Склад продуктів неповного згоряє горючих речовин складений і різноманітний. Це можуть бути горючі речовини:

- H, C, CH;
- атомарний водень і кисень;
- різні радикали – BIH, CH .

Продуктами неповного згоряє можуть бути також оксиди азоту, спирти, альдегіди, кетони і високотоксичні з'єднання, наприклад, синильна кислота.

Для захисту персоналу від дій небезпечних і шкідливих чинників пожежі проектом передбачено застосування промислового фільтруючого протигаза з коробкою марки В (жовтий).

До системи запобігання пожежі відносяться: запобігання утворення горючого середовища і освіти в горючому середовищі джерел запалення, забезпечення пожежебеспеки устаткування.

Щоб запобігти пожежі в обчислювальних центрах, проектом пропонується виконання наступних вимог:

- електроживлення ЕОМ має автоматичне блокування відключення електроенергії на випадок перегріву системи, що може бути результатом зупинки системи охолоджування і кондиціонування;
- система вентиляції обчислювальних центрів обладнується блокуючими пристроями, що забезпечують її відключення на випадок пожежі. Система обладнується вогнеперегороджуючими клапанами;
- застосування устаткування, що задовольняє вимогам електростатичної іскробезпеки [7];
- після закінчення роботи, перед закриттям приміщення, всі електроустановки і персональні комп'ютери відключаються від сіті електроживлення;
- в приміщеннях обчислювальних центрів забороняється:
  1) влаштовувати електророзетки на основах, що згоряють;
  2) використовувати синтетичні доріжки і килими;
  3) користуватися побутовими електронагрівальними приладами;
  4) захаращувати евакуаційні виходи і проходи;

5) влаштовувати на вікнах глухі грати;
6) залишати без нагляду включену в електромережу апаратуру, що використовується для вимірювань і нагляду.

Для протипожежного захисту проектом пропонується обладнати приміщення площею 18 м2, яке відноситься до категорії В, автоматичною протипожежною сигналізацією із застосуванням датчиків сповіщення РІД-1 (оповіщувач димовий іонізаційний) в кількості 1 штуки і застосовується в первинних засобах пожежегасінні. Площа контрольована оповіщувачем 150 м2.

Крім того, необхідно проводити навчання робочого персоналу правилам пожежної безпеки.

Розрахуємо вірогідність виникнення пожежі у виробничому приміщенні у разі запалювання транзистора:

$$Q = л \cdot T \cdot Ркз/отк \cdot Qвоспл \cdot Рзащ \qquad (4.8)$$

де л – інтенсивність відмов пожежеопасних ЕРІ;

Т – час роботи пожежеопасного ЕРІ за оцінюваний інтервал часу;

Ркз/отк - умовна вірогідність виходу ЕРІ в стан короткого замикання при його відмові;

Qвоспл - вірогідність запалювання ЕРІ, що знаходиться в стані короткого замикання;

Рзащ – вірогідність відмови захисту пожежеопасного ЕРІ. Якщо захист відсутній, Рзащ приймається рівній 1.

Вірогідність виникнення пожежі у разі запалювання транзистора:

$$Q = 1 \cdot 10^{-6} \cdot 1 \cdot 10^{-4} \cdot 0.1 \cdot 1 \cdot 10^{-4} = 1 \cdot 10^{-15}$$

Розрахована вірогідність виникнення пожежі значно менше допустимої, яка складає $1*10^{-6}$.

В даному розділі були проаналізовані небезпечні і шкідливі виробничі чинники, що роблять вплив на персонал, розроблені заходи щодо техніки безпеки, заходу, забезпечуючи виробничу санітарію і гігієну праці, а також заходи щодо пожежної профілактики.

ВИСНОВКИ

У дипломній роботі бакалавра була поставлена задача створення трьох модульної системи відстеження, збору, зберігання та відображення необхідної користувачеві інформації. Систему було створено, на прикладі збирання та аналізу геометок на карті Google. Подальша модернізація системи приведе до створіння універсального SDK, яке дозволить користувачу (програмісту) створювати власну систему для збору власно йому необхідної інформації.

Під час виконання розділу «Охорона праці» було проаналізовано робоче місце програміста на відповідність до встановлених норм. У ході аналізу умов праці було виявлено підвищений рівень шуму. У ході розрахунків було встановлено, що рівень шуму не перевищує допустимі значення. Додаткові заходи по покращенню умов праці не потрібні.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1) Харди Б, Филлипс Б. Программирование под Android. Для проффесионалов. [Текст] / Б. Харди, Б. Филлипс - Санкт-Петербург .: Питер, 2014 - 592 с.

2) Экель Б., Филлософия Java. Библиотека программиста. 4-е изд. [Текст] / Б. Экель - Санкт-Петербург .: Питер, 2009 - 640 с.

3) WebSocket. Реализация веб приложения с ипспользованием Jetty Web Socket. Часть 1. [Электронный ресурс] - Режим доступа: www/ URL: http://habrahabr.ru/post/128380/ - 15.05.2015 - Загл. с экрана.

4) Введение в JSON [Электронный ресурс] - Режим доступа: www/ URL: http://www.json.org/json-ru.html - 15.05.2015 - Загл. с экрана.

5) Rick Rogers John Lombardo, Android Application Development [Text] / Rick Rogers John Lombardo; - ECOM Pablysherz, ISBN 978-5-9790-0113-5, 978-0-596-52147-9;

6) ГОСТ 12.1.005–88. «ССБТ. Общие санитарно–гигиенические требования к воздуху рабочей зоны».

7) ГОСТ 12.0.003–74. «ССБТ. Опасные и вредные производственные факторы. Классификация.»

8) ГОСТ ГОСТ 12.1.030-81 «ССБТ. Электробезопасность.Защитное заземление. Зануление»

9) ГОСТ 12.1.003-83. «ССБТ. Шум. Общие требования безопасности»

10) ДБН В.2.5-28-2006. «Природне і штучне освітлення»

11) НАПБ Б.03.002-2007. «Нормы определения категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности»

12) ДСТУ Б А.3.2-13:2011 «Система стандартів безпеки праці. Будівництво. Електробезпечність. Загальні вимоги»

ДОДАТОК А.
Лістинг коду

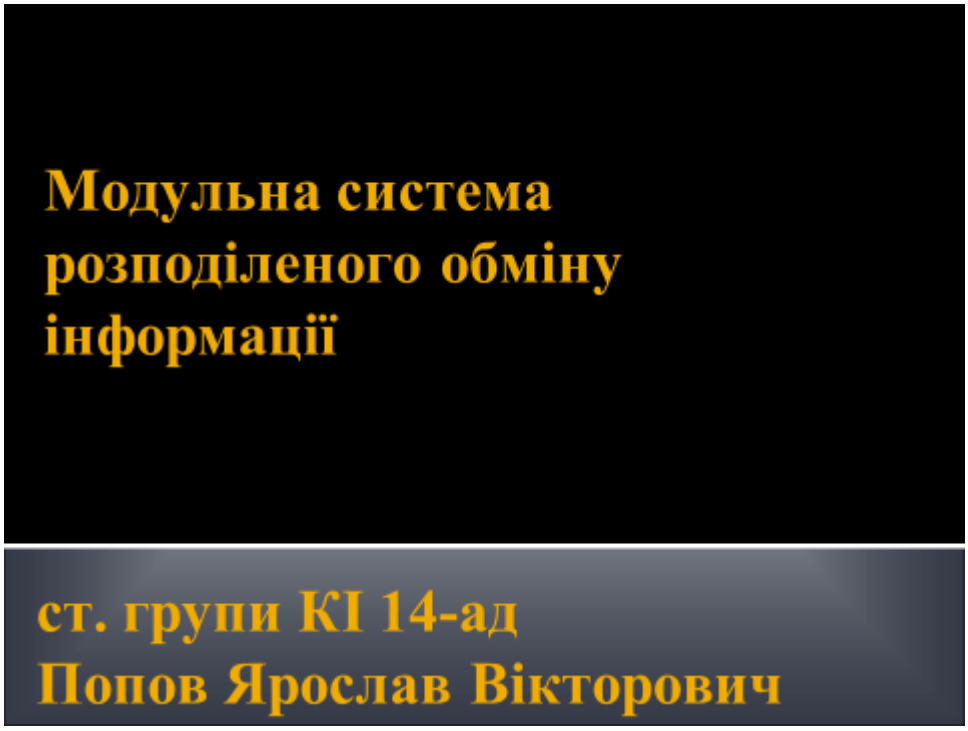
```
package com.stfalcon.websocket;

import android.util.Log;


import com.neovisionaries.ws.client.WebSocket;
import com.neovisionaries.ws.client.WebSocketAdapter;
import com.neovisionaries.ws.client.WebSocketException;
import com.neovisionaries.ws.client.WebSocketFactory;
import com.neovisionaries.ws.client.WebSocketFrame;


import java.io.IOException;
import java.security.NoSuchAlgorithmException;
import java.util.List;
import java.util.Map;


import javax.net.ssl.SSLContext;




public class ClientWebSocket {


    private static final String TAG = "Websocket";
    private MessageListener listener;
    private String host;
    private WebSocket ws;




    public ClientWebSocket(MessageListener listener, String host) {
        this.listener = listener;
        this.host = host;
    }


    public void connect() {
        new Thread(() -> {
```

```
        if (ws != null) {
            reconnect();
        } else {
            try {
                WebSocketFactory factory = new WebSocketFactory();
                SSLContext context = NaiveSSLContext.getInstance("TLS");
                factory.setSSLContext(context);
                ws = factory.createSocket(host);
                ws.addListener(new SocketListener());
                ws.connect();
            } catch (WebSocketException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
        }
    }).start();
}


private void reconnect() {
    try {
        ws = ws.recreate().connect();
    } catch (WebSocketException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}


public WebSocket getConnection() {
    return ws;
}


public void close() {
    ws.disconnect();
}




public class SocketListener extends WebSocketAdapter {
```

```
        @Override
        public void onConnected(WebSocket websocket, Map<String, List<String>> headers)
throws Exception {
            super.onConnected(websocket, headers);
            Log.i(TAG, "onConnected");
        }


        public void onTextMessage(WebSocket websocket, String message) {
            listener.onSocketMessage(message);
            Log.i(TAG, "Message --> " + message);
        }


        @Override
        public void onError(WebSocket websocket, WebSocketException cause) {
            Log.i(TAG, "Error -->" + cause.getMessage());


            reconnect();
        }


        @Override
        public void onDisconnected(WebSocket websocket,
                                   WebSocketFrame serverCloseFrame, WebSocketFrame
clientCloseFrame,
                                   boolean closedByServer) {
            Log.i(TAG, "onDisconnected");
            if (closedByServer) {
                reconnect();
            }
        }


        @Override
        public void onUnexpectedError(WebSocket websocket, WebSocketException cause) {
            Log.i(TAG, "Error -->" + cause.getMessage());
            reconnect();
        }


        @Override
        public void onPongFrame(WebSocket websocket, WebSocketFrame frame) throws
Exception {
            super.onPongFrame(websocket, frame);
            websocket.sendPing("Are you there?");
        }
    }
```

```
    public interface MessageListener {
        void onSocketMessage(String message);
    }
}
```

ДОДАТОК Б.
Електронні плакати



Ситуація, коли необхідно вчасно і якісно реагувати на зміни в інформаційному середовищі, отримувати потрібний потік інформації і реагувати тільки на необхідні відомості стає все більш актуальною. Тому і виникає потреба в системі, яка відстежує тільки ту інформацію, яка необхідна конкретному користувачеві. Ця система повинна бути проста і універсальна, щоб вона могла б застосовуватися з абсолютно різними наборами даних, виконуючи основні функції щодо відстеження, збирання, зберігання та відображення інформації, а також повинна бути досить легковажною, для розміщення в будь-якому пристрої, від найменшого датчика до потужного ПК.

Постановка задачі

Метою даної роботи є створення такої системи, що складається з трьох частин.

Перша частина - модуль-сенсор, який буде виконувати збір даних і відсилати їх сервера. Цей модуль буде виконаний як додаток на Android.

Друга частина - сервер, який буде приймати дані від модулів-сенсорів, які будуть розміщені на безлічі Android пристроїв, збирати ці дані і зберігати їх в базі даних.

Третя частина - модуль аналізу і відображення зібраної інформації, також Android додатком.

Зібрані дані будуть зберігається в базі даних в хмарному сховищі.

Огляд необхідних технологій

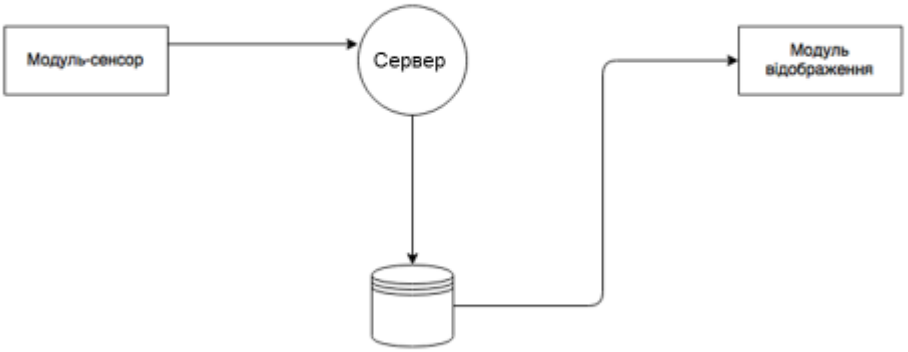
Для розробки були використані інтегровані середовища розробки:
- Android Studio? - Jet Brains Intellij Idea

Мова програмування JAVA:
- JDK
- Android SDK

технології:
-WebSocket

Сховище даних
-Parse хмара

Концептуальна схема системи



Особливості проектованої системи

- Простота - кожен модуль виконує одну елементарну задачу
- Універсальність - можливість працювати з будь-якими даними
- Легковажність і швидкодія - забезпечуються простотою системи.

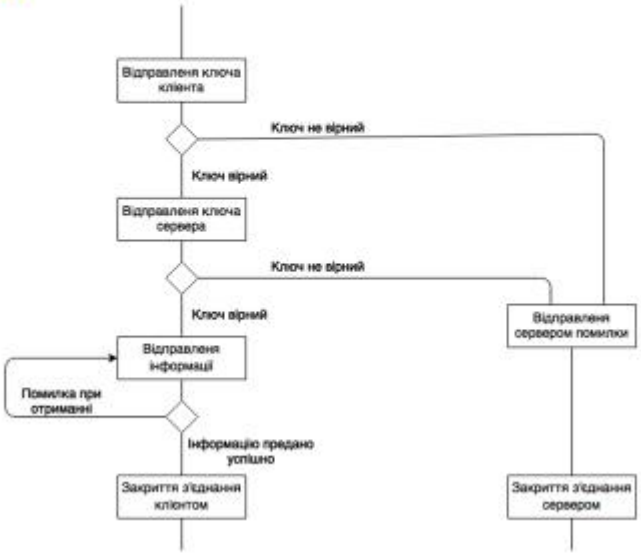
Взаємодія сенсор - сервер

Взаємодія сенсора з сервером полягає в єдиній функції - передачі даних.

Необхідно ввести ідентифікацію за двома ключам, щоб виключити втрату даних.

Необхідно розробити протокол спілкування сенсорів з сервером для збільшення надійності системи.

Протокол спілкування сенсора з серер



Формат переданих даних

Передача даних здійснюється у форматі JSON.

Структура повідомлення :?
{"Action": <тип дії>,? "Data": <дані>}

приклад:

{"Action": "GIVEN_CLIENT_KEY",? "Data":
"SOME_GENERATED_CLIENT_KEY"}

Формат переданих даних

Структура одного елемента набору даних.
{"Property_name": "<імя_елемента>",
"Property_type": "<тіп_елемента>",
"Property_value": "<значеніе_елемента>"}

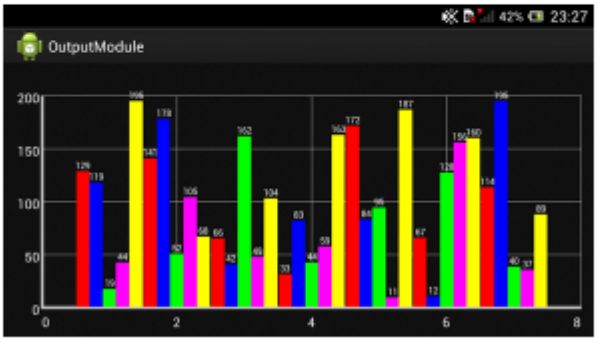
Структура цілого набору даних.

[Element, element, element, ..., element]:

[{"Property_name": "<імя_елемента>",
"Property_type": "<тіп_елемента>",
"Property_value": "<значеніе_елемента>"},
...,
{"Property_name": "<імя_елемента>",
"Property_type": "<тіп_елемента>",
"Property_value": "<значеніе_елемента>"}]

Кінцевий модуль

Демонстрація роботи кінцевого модуля відображена на малюнку



Охорона праці

При виконанні розділу «Охорона праці» були визначені:

- небезпечні і шкідливі фактори,
- розроблені організаційно-технічні засоби їх усунення
- Визначено можливі причини пожежі в приміщенні
- Розроблено план евакуації при пожежі
- Виконано розрахунок рівня шуму в приміщенні
- Розглянуто питання виробничої санітарії та промислової безпеки

Висновки

У кваліфікаційній роботі була і розроблена система збирає, зберігає, обробляє і відображає необхідні користувачу дані. В ході розробки були отримані навички проектування багатомодульних систем, парсинга JSON, програмування на Java під Android.

Вибрані технології:

- Java SDK;
- WebSocket;
- Хмарне сховище Parse.