ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра комп‘ютерної інженерії

Пояснювальна записка

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА СИСТЕМИ ДЛЯ ЗАБЕЗПЕЧЕННЯ КОНТРОЛЮ ТРАНСПОРТНИХ ЗАСОБІВ МОВОЮ C#»**

Виконав: студент 4 курсу, групи ПД–42
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Добрівський О.В
(прізвище та ініціали)
Керівник Трінтіна Н. А.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Телекомунікацій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Напрям підготовки -121 – "Інженерія програмного забезпечення"

ЗАТВЕРДЖУЮ

Завідувач кафедри
Інженерії програмного забезпечення
_____ О.В Негоденко
"____"_____2022 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Добрівський Олександр Валерійович
(прізвище, ім'я, по батькові)

1.Тема роботи: «Розробка системи для забезпечення контролю транспортних засобів мовою C#»
Керівник роботи Трінтіна Наталія Альбертівна канд.техн.наук, доц.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "__"___2022 року №__.
2. Строк подання студентом роботи
3. Вхідні дані до роботи: Інструкція до пристрою спостереження за рухомими об'єктами BI 910 TREK, документація протоколу Bitrek, різноманітна документація на мови C# та PHP, веб-сервер

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
Вступ
Аналіз предметної області
Обґрунтування вибору програмних засобів
Аналіз роботи протоколів
Опис реалізації проекту
Висновки
5. Перелік графічного матеріалу
1. Деякі діаграми контекстної моделі
2. Діаграма послідовностей
3. UML діаграма класів
4. Приклади роботи панелі
6. Дата видачі завдання 11 квітня 2022 р.

КАЛЕНДАРНИЙ ПЛАН РОБІТ

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	До 08.03.2020	виконано
2	Оформлення планування робіт	До 20.03.2020	виконано
3	Проведення аналізу предметної області	До 01.04.2020	виконано
4	Проведення структурно-функціонального моделювання процесів	До 28.04.2020	виконано
5	Розробка проекту	До 12.05.2020	виконано
6	Вступ, висновки, реферат	До 16.05.2020	виконано
7	Здача пояснювальної записки та файлів розробленого проекту	До 07.06.2020	

Студент _____ Добрівський О.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Трінтіна Н. А.
підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 61 с., 2 табл., 16 рис., 2 дод., 18 джерела.

ТЕХНОЛОГІЯ GPS, ПРОТОКОЛИ GPS-ТРЕКЕРІВ, C#, PHP, ПАНЕЛЬ КЕРУВАННЯ

Об`єкт дослідження –.використання технологій GPS для моніторингу транспортом.

Предмет дослідження – комп'ютерні системи та додатки для моніторингу та керування транспортом.

Мета роботи – створити зручний та універсальний додаток для кінцевого користувача, на базі сучасних технологій, для моніторингу транспортом.

Методи дослідження – методи проектування, перегляд та вивчення документації, перегляд наявних технологій.

Здійснено аналіз існуючих систем GPS; проаналізовані технології моніторингу транспорту із здійсненням gps пристроїв; досліджено протоколи за якими працюють трекери; реалізовано алгоритми та UML-діаграми; проаналізовано та опротестовано результати та зручність роботи створеної системи.

На основі результатів виконаних досліджень розроблено універсальний модуль зв'язку для прийому та передачі пакетів на виділений сервер.

ЗМІСТ

ВСТУП

Розвиток автомобілебудування обумовлює збільшення автомобільного транспорту як засобу сполучення, що використовується як для перевезення людей, так і вантажів різних габаритів, складу і призначення. Розвиток транспортних компаній визначається не тільки збільшенням парку транспортних засобів, а й впровадженням все більш сучасних систем управління. Для ефективного управління - як ефективного, так і економічного - необхідні сучасні логістичні системи, що надають швидку та вичерпну інформацію про керовані транспортні засоби. Поточний розвиток телекомунікацій і радіозв'язку, а також методів IT дозволяє впроваджувати комплексні послуги, що полягають в безперервному визначенні місця розташування транспортних засобів і автоматичному контролі за перевезеннями у внутрішніх та міжнародних перевезеннях (автомобільних і залізничних).

Ефективна організація перевезень та можливість своєчасної підготовки розвантаження вантажу, та швидке реагування на похибки в планових перевезеннях є важливими не тільки з міркувань безпеки, але й за рахунок оптимального використання рухомого складу і людського потенціалу.

Системи моніторингу транспортних засобів можуть мати велике значення у випадку перевезення цінних вантажів або при перевезеннях які потребують особливого контролю з інших причин (наприклад, стратегічних). У зв'язку з цим до класу користувачів можуть відноситися митні служби, поліція, прикордонна служба і охоронні підприємства, що спеціалізуються наприклад на перевезенні грошей, спеціальних вантажів (радіоактивних матеріалів), вугілля, та інше. Серйозними одержувачами систем моніторингу можуть виявитися власники транспортних флотів і вантажовідправники.

Метою дипломної роботи являється розробка програмного засобу, який дозволяє проводити моніторинг транспортних засобів, відображати місцезнаходження об’єкту, пройдений шлях. Та надав би можливість оброблювати пакети від різних gps-трекерів.

Для розробки програмного забезпечення системи моніторингу були використані мови програмування C# та PHP. Розробка графічного інтерфейсу користувача створювалися за допомоги HTML5, Java Script та CSS

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Глобальна навігаційна система – GPS

GPS (Global Positioning System) - американська супутникова навігаційна система, що покриває всю земну кулю. Він служить, серед іншого для визначення географічного положення (довгота, широта і еліпсоїдальна висота). Система загальнодоступна, її послугами може скористатися будь-який бажаючий, досить мати відповідний GPS-приймач.

Технологія GPS [1] була створена Міністерством оборони США і є результатом досвіду, отриманого при створенні і використанні супутникової навігаційної системи TRANSIT, особливо супутників серії TIMATION. З 1960 р. TRANSIT успішно пройшов випробування Департаментом ВМС США (US Navy), а випробування нової системи GPS почалися в 1972 році. Перший супутник Block I SVN 1, був запущений 22 лютого 1978 р. Спочатку система використовувалася тільки для військових цілей і була повинна відповідати ряду вимог: здатність визначати позицію в режимі реального часу, незалежно від умов в яких використовується система, повинна бути стійкість до випадковим та навмисним збурень, точність наведення ракети на ціль-мішень 5 м, низька ціна приймача, доступність по всьому світу,

На сьогоднішній день на орбіту послідовно виведено 6 поколінь супутників. Це були послідовно запущені супутники I блоку (SVN1 - SVN11), супутники II блоку (SVN13 - SVN21) - більше не працюють, супутники блоку IIA (SVN22 - SVN40), супутники блоку IIR (SVN41 - SVN62), супутники блок IIP-M і супутники новітнього покоління блоку IIF (перший запущений 28 травня 2010 р.) термін служби супутників останнього покоління оцінюється приблизно в 12 років. Супутники що утворюють так званий космічний сегмент системи, розташовуються на кругових орбітах з нахилом 55° (Блок IIA, IIR, IIR-M) або 63° (Блок I) щодо площини екватора, на висоті 20183 км. Обліт навколо Землі супутники роблять за 11 годин 58 хвилин (половина зоряної доби).

На орбіті одночасно знаходиться 30 діючих супутників, роботу яких контролює так званий наземний сегмент системи. Він складається з 12 станцій, які розташовані максимально рівномірно по екватору, таким чином, щоб кожен із супутників було постійно видно як мінімум з двох станцій. Основна станція спостереження розташована на авіабазі Shriever AFB (раніше Falcon) в Колорадо-Спрінгс (США). Інші станції спостереження ВПС США розташовані на Гаваях, мисі Канаверал, острові Вознесіння та острові Дієго-Гарсія і атолі Кваджалейн. Шість станцій знаходяться у віданні NGA (Національне агентство геопросторової розвідки), це станції у Вашингтоні, Еквадорі, Аргентині, Лондоні, Бахрейні та Австралії. На основі їх спостережень розраховуються нові параметри орбіт супутників.

Принцип роботи системи GPS полягає в точному вимірі часу і зчитуванні положення супутників на орбіті (кожен із супутників оснащений атомним годинником) та посилає сигнал на двох несучих частотах f1 = 1575,42 МГц (довжина хвилі 19,029 см) і f2 = 1227,6 МГц (довжина хвилі 24,421 см). Це дає можливість вимірювати відстань між супутниками та приймачем двома методами: кодовим і фазовим. На практиці для визначення положення в тривимірному просторі і часі системі необхідно одночасно приймати сигнали як мінімум з чотирьох супутників. Приймач користувача обчислює три псевдо-дальності до супутників та похибку у часі (різницю між недостатньо точним кварцовим годинником, встановленим на приймачі, і точними атомними годинами на супутнику). Потім точні координати супутника транслюються в навігаційному повідомленні.

Системі GPS має два рівні доступу. Прецизійний доступ призначений для Збройних сил США і союзних сил та стандартний доступ, з меншою точністю, який використовується цивільними. Спочатку він був навмисно порушений псевдовипадковою помилкою, яка унеможливлювала отримання більш точних координат ніж 100 м., якщо тільки не використовувалося усереднення показань, проводячи тривалий стаціонарний відлік. Механізм глушіння сигналу був

відключений 1 травня 2000 року при президенті Біллі Клінтоні, що підвищило точність позиціонування для цивільних користувачів приблизно до 4-12 метрів.

Також є додатковий спосіб який дозволяє визначити більш точне положення за допомогою GPS, це вимірювання диференціальної глобальної системи позиціонування (DGPS). У цьому методі використовуються так звана Базова (опорна) станція, тобто приймач, розташований в точно визначеній точці. Який визначає поточні диференціальні поправки для окремих супутників, усуваючи помилки годинників супутників, ефемериди і затримки сигналів, що виникають в результаті взаємодії іоносфери і тропосфери. Для роботи диференціальної системи DGPS необхідна можливість відправки поправок на другий мобільний приймач, наприклад, через VHF (Very High Frequency, тобто УКХ-радіохвилі з частотою від 30 до 300 МГц) або GPRS (General Packet Radio Service — пакетна передача даних в технології зв'язку операторів мобільного зв'язку стандарту GSM). Дану процедуру можна використовувати як в режимі реального часу при проведенні вимірювань, так і при подальшій обробці зібраних даних, так звана постобробка. Використання методу DGPS дозволяє досягти точності 0,5-2 м і вище.

GPS-сегменти:

1. Космічний. До його складу входять 24 супутники (плюс кілька резервних) типу «Навстар», що рухаються по шести рівномірно розташованим круговим орбітам, чотирьом на рівних відстанях, з часом обертання 12 годин, на висоті 20 200 км, з нахилом 55 ° до екватора. Завжди видно від 5 до 12 супутників.

2. Контроль та моніторинг. Сегмент управління складається з головної станції управління (MCS), розташованої на базі ВПС Колорадо-Спрінгс, США. Станції моніторингу розташовані, в тому числі на Гаваях, Мис Канаверал, Острів Вознесіння. Наземні антени.

3. Користувацький. Технологія GPS знаходить все більше застосувань, звідси і користувальницький сегмент цієї системи - призначений для користувача. Приймачі істотно розрізняються по технічній досконалості і пропонованим функціям. Сучасні GPS-приймачі зазвичай працюють на тридцяти

незалежних каналах, кожен з яких пристосований для прийому і обробки сигналів з одного супутника, причому процеси прийому та обробки сигналів здійснюються в такому багатоканальному приймачі одночасно. Старі приймачі працюють на 8 або 12 каналах, що однак, досить - в Україні, як правило, одночасно видно близько 10-12 супутників.

На рис.1 проілюстровано ключові сегменти GPS.

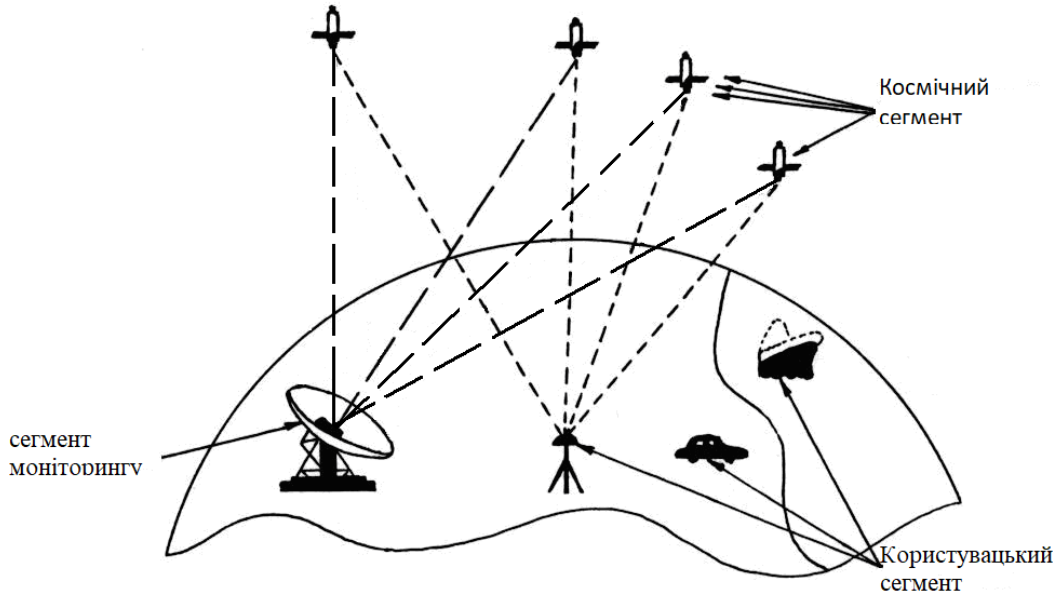


Рисунок 1 - ключові сегменти GPS

Хоча GPS в даний час є найбільш використовуваною супутниковою навігаційною системою, в даний час існують або знаходяться в стадії розробки інші системи типу GPS . До них відносяться російська система ГЛОНАСС, китайська система BeiDou і європейська система Galileo.

Система ГЛОНАСС є другою в світі глобальною навігаційною супутниковою системою. На сьогодні система ГЛОНАСС налічує 24 діючих супутника, що всього на два менше від необхідної кількості для глобального покриття. ГЛОНАСС став загальнодоступним в 2007 році і в даний час активно просувається на Близькому Сході, в Східній Європі, Індії та Південній Америці. ГЛОНАСС можна використовувати в поєднанні з GPS, що збільшує кількість

видимих супутників і, отже, може забезпечити підвищену точність в густонаселених міських районах.

Китай також розробила власну незалежну глобальну навігаційну супутникову систему BeiDou. Незважаючи на те, що він орієнтований на військову службу, він також має цивільне призначення. В 2020 році система була повністю введена в експлуатацію і є сумісна з іншими глобальними навігаційними супутниковими системами. Поточна кількість супутників сягає 40 з них для повного угруповання потрібно лише 35 супутників.

Система Galileo Європейського Союзу являється незалежною системою під цивільним контролем і не підпорядковується військовим. Galileo в даний час знаходиться в стадії розробки, і як очікується буде частково введений в експлуатацію в 2025 році.

1.2 Система моніторингу та контролю транспортних засобів

GPS-моніторинг — інструмент, який дозволяє контролювати розташування транспортних засобів та їх параметри роботи за допомогою спеціальних пристроїв що називаються GPS-контролерами. Система моніторингу транспортних засобів складається з бортового обладнання (GPS-контролера з датчиками) та програмного забезпечення системи моніторингу, за допомогою якого здійснюється GPS-моніторинг і складаються необхідні звіти. В залежності від можливостей, які пропонуються конкретними рішеннями, можна збирати дані про розташування, маршрути та швидкість транспортних засобів, а також про продуктивність автомобіля (напруга батареї, відкриття дверей, швидкість двигуна, повідомлення про помилки) та збір і аналіз отриманої інформації.

Головний контролер який встановлюється в транспортний засіб має модульну структуру. Він відповідає за доставку та отримання центром моніторингу інформації про місцезнаходження транспорту та стану на зафіксовані події. В основі пристрою лежить багатофункціональна мікропроцесорна система, інтегрована з дванадцяти канальним GPS-приймачем

і модулем зв'язку GSM/GPRS. Модулі взаємодіють один з одним по протоколу RS485.

Основною метою системи є визначення положення транспортного засобу в будь-якій точці на Землі, шляхом вказівки географічних координат, які чітко визначають його місце розташування. Для досягнення цієї мети зазвичай використовується супутникова система GPS, яка була розглянута в попередньому розділі 1.1. Проте найбільшою перевагою використання GPS-моніторингу є економія експлуатаційних витрат та економія часу. Точний аналіз маршрутів, звичок водіїв або економічності автомобілів дозволяє оптимізувати роботу автопарку. Ще однією з переваг є більш висока ефективність, за допомоги звітності та аналізу середньої швидкості транспортних засобів, того, як водії прискорюються та скільки часу вони проводять на стоянці з не виключеним двигуном, можуть привести до зниження витрати палива, і в свою чергу, до більшої фінансової економії. Навіть маленькі зміни у великій кількості автомобілів, у довгостроковій перспективі можуть дати дуже хороші результати.

На рис. 2 показана загальна (спрощена) блок-схема системи моніторингу, що показує, як вона працює. Вона включає, зокрема, наземну частину, це центр управління який підключений до онлайнового терміналу і бортової диспетчерської станції, що працює з серверами по локальній мережі. Це дозволяє отримувати звіти, а також відправляти SMS-повідомлення в бортовий модуль і отримувати SMS-повідомлення в зворотному напрямку.

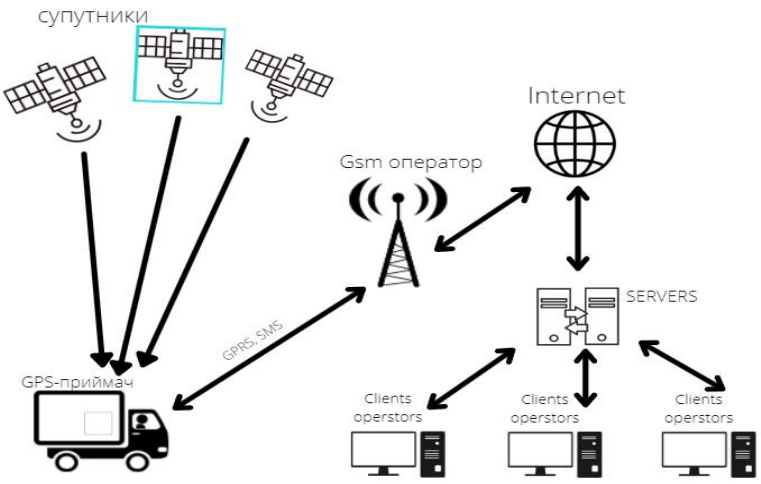


Рисунок 2. Блок-схема системи

Структура алгоритму зв'язку GPS-приймача (Gps-трекер) [2] із системою моніторингу (диспетчерським центром) представлена на рис. 3. На йому представлені три основні блоки : система моніторингу (диспетчерський центр), GSM станція, GPS-приймач (транспортний засіб).

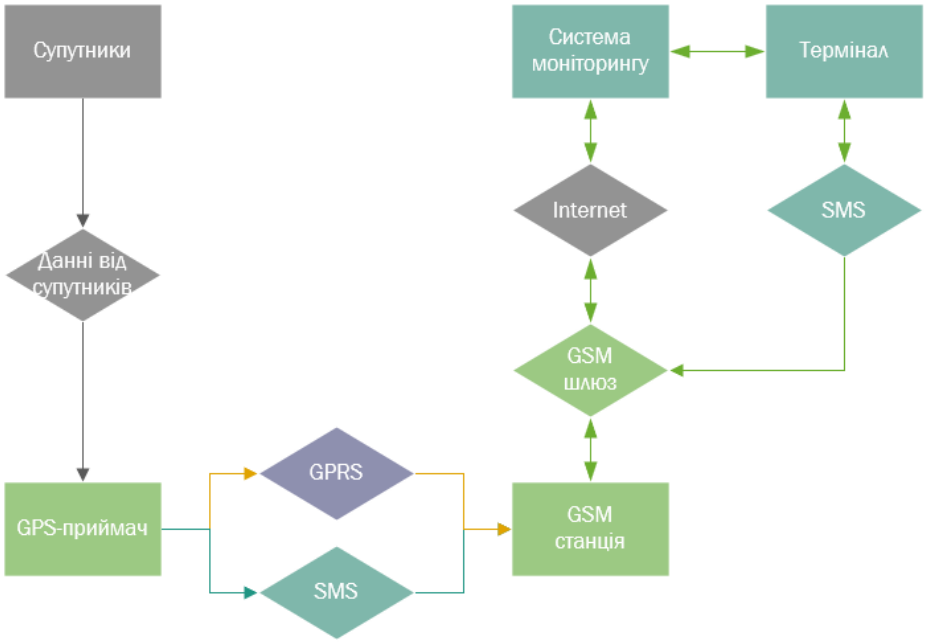


Рисунок 3. Алгоритм зв'язку транспортного об'єкта з системою моніторингу (диспетчерським центром)

Для встановлення зв'язку з транспортним засобом, система моніторингу використовує Інтернет або SMS використовуючи для цього спеціального термінал який у совою чергу зв’язується з GSM-станцією оператора. Потім після отримання даних транспортний засіб (оснащений GPS-приймачем) відправляє необхідну інформацію на GSM-станцію двома альтернативними маршрутами: через GPRS або SMS. Далі станція GSM відправляє зібрані дані до центру моніторингу.

Завдяки модульній структурі конфігурації системи, обсяг її функціональних можливостей легко адаптується до вимог і потреб споживача. Базовими елементами системи є підсистема локалізації об'єкта та центр моніторингу.

Підсистема локалізації об'єкта заснована на пристроях моніторингу, які визначають положення контрольованого об'єкта за допомогою GPS-приймача, котрий із заданою періодичністю передає інформацію в центр спостереження по мережі GSM, використовуючи технології GPRS і SMS. Доступ до диспетчерського центру може здійснюватися через комп'ютерну мережу (Інтернет). Після авторизації через користувацький інтерфейс споживач отримує доступ до даних і місцезнаходження контрольованих об'єктів.

При використанні GPRS, канал передачі виділяється за вимогою, а за допомоги мультиплексування створюється можливість використовувати один канал декільком користувачам. Таке рішення значно знижує експлуатаційні витрати системи, та гарантує практично необмежену функціональність. Коли GSM-оператор не в змозі надати ресурси для передачі через GPRS (в основному малоурбанізовані райони), пристрій автоматично перемикається в режим передачі коротких текстових повідомлень, тобто SMS.

Частота відправки інформації від GPS-трекера [2] може бути змінена від потреб користувача в центрі моніторингу. В разі відсутності можливості передачі даних, система здатна запам'ятовувати інформацію про пройдений маршрут, кількість і тривалість зупинок, об’єми витраченого палива і т. ін. в додатково встановлену флеш-пам’ять, і відправляти їх в центр спостереження, як тільки зв'язок буде відновлений.

Обсяг даних, що надсилаються з GPS-пристрою в транспортному засобі в центр спостереження, включає:

- географічні координати об'єкта разом з його висотою над рівнем моря;
- миттєва швидкість рухомого об'єкта;
- дата і час вимірювання;
- якість і рівень сигналу;
- кількість "видимих" супутників GPS (потрібно не менше 3-х супутників);
- стан приймача GPS.

З метою підвищення безпеки роботи системи передача шифрується, а вся інформація про місцезнаходження транспортних засобів фіксується в базі даних системи.

База даних збирає, серед іншого:

- інформацію про напрямки руху та маршрут об'єктів в будь-який період часу;
- реєстраційні дані контрольованих об'єктів;
- сигнали та тривоги, що надходять від об'єктів;
- дані про менеджерів;
- дані про витрати на обслуговування транспортний засобів.

На багатошаровій цифровій карті відображається як поточне положення контрольованих об'єктів, так і історія їх маршрутів. Разом з інтегрованою базою даних це основний інструмент оператора центру спостереження.

1.3 Клієнт-серверна архітектура.

Програмне забезпечення для моніторингу транспортом являє собою клієнт-серверний додаток який побудований [3] на трирівневій архітектурі. На сьогодні ця архітектура є домінуючою для програмного забезпечення в клієнт-серверних додатках.

Трирівнева Архітектура (Three-Tier architecture або three-layer architecture) - клієнт-серверна архітектура, [4] в якій користувальницький інтерфейс, обробка

даних і зберігання даних розробляються у вигляді окремих модулів, зазвичай на окремих платформах; концепція трирівневої архітектури походить від Rational Software.

Архітектура цього типу дозволяє оновлювати або замінювати окремі модулі незалежно один від одного в міру зміни технічних умов-наприклад, зміна операційної системи на вашому комп'ютері (наприклад, з Windows на Linux або навпаки) впливає тільки на рівень користувальницького інтерфейсу, але не на обробку і зберігання даних.

Обробка даних на сервері додатків також може складатися з декількох окремих шарів, в результаті чого така архітектура перетворюється в багаторівневу архітектуру.

В трирівневій архітектурі додатки розділені на три рівня:

Рівень представлення являє собою інтерфейс користувача та комунікаційний рівень програми, на якому кінцевий користувач взаємодіє з додатком. Основним завданням цього рівня є відображення даних та збір інформації від користувача. Цей шар верхнього рівня може працювати у веб-браузері, як офісний додаток або як графічний інтерфейс користувача (GUI). Цей рівень представлення наприклад у веб-додатку розробляються за допомогою HTML5, JS та CSS або фреймворках на їх основі.

Прикладний рівень - проміжний рівень на якому зазвичай проводиться обчислення та логіка основної програми. Саме тут обробляється інформація, наприклад шляхом порівняння інформації яка була раніше зібрана на рівні представлення з іншою інформацією, з рівня даних, використовуючи для цього бізнес-логіку, тобто певний набір бізнес-правил. На цьому рівні інформація може додаватися, видалятися, змінюватися на інше. Зазвичай [8] прикладний рівень розробляється з використанням таких язиків як PHP, Python, Perl, Java.

Рівень управління даними. Цей рівень зберігає та управляє інформацією, що використовується додатком. Прикладом таких систем є СКРБД (система управління реляційною базою даних) такі як MySQL, MariaDB, OracleDB або

MSSQL Server, також можуть використовуватися нереляційні бази даних NoSQL, такі як MongoDB.

Основною перевагою трирівневої архітектури є логічний і фізичний поділ функцій. Кожен рівень може працювати на різних операційних системах і серверних платформах — наприклад, веб-сервері, сервері додатків, сервері бази даних — які найкраще відповідають функціональним вимогам. Кожен рівень працює на одному або декількох виділених фізичних або віртуальних серверах, тому служби кожного рівня можна налаштовувати і оптимізувати, не зачіпаючи інші рівні.

Інші переваги (у порівнянні з однорівневою або дворівневою архітектурою):

Прискорена розробка: оскільки різні рівні можуть розроблятися одночасно різними командами, організація може швидше виводити додаток на ринок, а розробники можуть використовувати новітні мови та інструменти, адаптовані для кожного рівня.

Оптимізована масштабованість. При необхідності кожен фізичний рівень можна масштабувати незалежно від інших.

Підвищена надійність: збій одного рівня менше впливає на доступність або продуктивність інших рівнів.

Більш високий рівень безпеки: за рахунок того що рівень представлення та рівень даних не можуть безпосередньо взаємодіяти один з одним і як правило взаємодіють лише через прикладний рівень, зменшується ризик доступу до приватної інформації що зберігається у базах даних та запобігає іншим шахрайським діям.

2 ОБҐРУНТУВАННЯ ВИБОРУ ПРОГРАМНИХ ЗАСОБІВ

2.1 Вибір технології зберігання та обробки даних.

Так як у проекті потрібно зберігати велику кількість однотипних даних та її використовувати, з’явилася потреба вибору технології зберігання даних. Для цього використовуються технології баз даних, найпоширенішими типами баз даних являються нереляційні NoSQL та реляційні SQL.

Найбільш часто використовуваним типом баз даних є реляційні бази даних тобто, вони забезпечують найбільш надійний спосіб доступу до структурованої інформації. Реляційні бази даних називаються так тому, що вони зазвичай містять кілька таблиць, деякі з яких пов'язані один з одним. Кожен рядок в таблиці має свій унікальний ключ. Рядки в таблиці можуть бути пов'язані з рядками в інших таблицях за допомогою додаткового стовпця, що містить ключ зв'язаного рядка.

База даних NoSQL або нереляційна база даних дозволяє зберігати і маніпулювати неструктурованими і частково структурованими даними (на відміну від реляційних баз даних, які визначають, як організовані всі дані, що вводяться в базу даних). Нереляційні технології дуже добре підходять для зберігання різнотипних даних та для проектів які постійно змінюють напрямок розробки або пріоритети (наприклад розробка комп’ютерних ігор).

Для даного проекту було вибрано використання реляційних баз даних SQL, з причин великої кількості однотипної інформації яку потрібно структурувати для подальшої роботи с нею. Оскільки були обрана технологія SQL, то залишилося обрати ще систему управління для неї. Найбільш популярними [5] системами являються: Oracle Database, Microsoft SQL, MySQL, MariaDB.

Oracle Database – це корпоративна СКБД (Система Керування Базами Даних) яка була розроблена компанією Oracle у 70-х роках. Остання випущеною версією програмного продукту є 21c котра випущена у 2021 році. Активно підтримується розробниками та оновлюється. На даний час являється

найпоширенішою комерційною системою. Завдяки кросплатформеності може використовуватися в різних системах. Проте ця система нами не розглядається із-за її комерційної ліцензії.

Система Microsoft SQL розроблена компанією Microsoft у 80-х роках. Це система керування базами даних яка може працювати в хмарному середовищі так і на локальному сервері, або можливе одночасне комбінування обох способів [6] застосовування. Після того як Microsoft випустило версію MSSQL Server 2016, з'явилася можливість використання цього продукту в операційних системах Linux, раніше він працював на платформі Windows. Остання версія що випущена SQL Server 2019. У даному проекті система не розглядається з тих ще міркувань що й Oracle, він розповсюджується на комерційний основі.

MySQL - одна з найпопулярніших СКБД і найбільш часто використовуваних баз даних у веб-додатках, які працюють під управлінням операційної системи Linux. MySQL має безкоштовний пакет програм який постійно оновлюється, розширюючи функціонал і покращуючи безпеку. Існують спеціальні платні версії, призначені для комерційного використання. В безкоштовній версії увага приділяється швидкості та надійність, а не на повноту функціоналу. Виконується на операційних системах Linux (Unix), Windows та macOS. Остання версія була випущена у 2022 році та має відкриту ліцензію GNU GPL 2. Розглядається як основна СКБД, тому що має безкоштовне розповсюдження та добре показує себе при роботі з веб-додатками.

MariaDB ще одна безкоштовна СКБД яка має безліч доступних плагінів (розширень) одна із самих швидко оновлюємих систем контролю на даний момент. MariaDB це фактично відгалуження від СКБД MySQL, провідний розробник Майкл Віденіус являється автором оригінальної версії MySQL. Проте завдяки добавленій оптимізації, система стала більш швидкою та продуктивною СКБД. Остання версія 10.7.3 випущена у 2022 році. Підтримується системами Microsoft Windows, macOS, Linux, Solaris та OpenBSD. Ця система має багато переваг, проте на даний момент стабільність у неї нижча, ніж у MySQL.

Детальне порівняння систем контролю наведено у табл.1.

Таблиця 1 – Порівняльна таблиця засобів СКБД

СКБД Характеристика	Oracle DB	Microsoft SQL	MySQL	MariaDB
Операційні системи	Linux, Windows Solaris	UNIX, OS/2, Windows	Linux, Unix, macOS, Solaris, Windows	Linux, Unix, macOS, Solaris, Windows
Стабільність	+	+	+	+/-
Ліцензія	Комерційна	Комерційна	GNU GPL, Комерційна	GNU GPL
Використання	Мережеве			
Кластеризація	Підтримується			

При детальному аналізі систем контролю було прийнято рушення використати MySQL з причини, її кросплатформеності та безкоштовної ліцензії GNU GPL. Зокрема на вибір системи керування вплинула її стабільна роботи у веб-проектах.

2.2 Вибір мов програмування та інтегрованого середовища розробки.

У даному проекті було вирішено використовувати веб-технології для створення додатку.

Основними приводами для цього стали:

- Можливість безшовного оновлення системи, не чекаючи виходу нових версій програми;
- Доступність додатку, немає потреби встановлювати клієнтське програмне забезпечення;
- Зменшення собівартості обслуговування програмного забезпечення;

- Мультиплатформеність, можливість використання додатку у різних системах.

Для веб-розробки найчастіше використовують мови: HTML 5, каскадні таблиці стилів CSS 3, мову сценаріїв JavaScript, PHP та системи управління реляційними базами даних MySql. Проте це не єдині технології, а найбільш часто використовувані через їх можливості, розповсюдження та безкоштовній доступ.

Для роботи на стороні браузера, нам потрібен тільки сам веб-браузер та будь який редактор коду, щоб мати можливість створити веб-сторінку. Мови які використовуємо на стороні клієнта (браузера): HTML5, CSS, JS (JavaScript).

Якщо ми хочемо використовувати технологію PHP [7] та бази даних MYSQL, нам потрібен сервер, найбільш часто використовуваним сервером є APACHE (безкоштовний сервер), який необхідно встановити та налаштувати. Професійні сервери Apache налаштовуються в середовищі LINUX, але ще можливо встановлення сервера в систему Windows.

Тестовий сервер можливо легко створити на локальному комп'ютері, встановивши вручну всі пакети (інтерпретатор PHP, базу даних mysql та сервер Apache). Також існують спеціальні програми які вже містять необхідне вже встановлене програмне забезпечення, наприклад XAMP. Встановивши xamp, ми одразу отримуємо готове середовище для використання серверних мов програмування. Технології, включені в xamp (Apache, MariaDB або MySQl, PHP різних версій, phpMyAdmin, OpenSSL, підтримка Perl та ін. Після установки сервера і правильного налаштування ми можемо повною мірою використовувати переваги технологічних рішень на стороні сервера з комбінацією мов, що працюють на стороні браузера.

Для написання логіки та основних функцій продукту, тобто бекенду, було вирішено використовувати мову програмування PHP. Його було обрано із-за того що він являється одним із популярних мор розробки веб-додатків і має багату документацію. Також входить у пакет програм xamp, що полегшує процес розробки. Зокрема головним аргументом для вибору php стало те що, проекти

розроблені на цій мові програмування можуть бути легко перенесені практично на любий веб-сервер та не буде потребувати допоміжних налагоджень.

Однак для розробки модулю зв'язку було вирішено обрати мовою C#, яка добре себе показує при створенні додатків які працюють із технологією TCP. Пошук допоміжної мови розробки пов'язана с тим що технології PHP погано працюють з підняттям сесій та подальшого його утримання. Що для даного проекту являється однією із основних вимог, практично усе тримається навколо зв'язку, між трекером та сервером.

Для створювання додатків обраними мовами програмування можна використовувати технології Visual Studio та Visual Studio Code, NetBeans, IntelliJ IDEA, Eclipse.

Visual Studio – програмний продукт розроблений компанією Microsoft, який об'єднує у собі пакет програм який являє собою IDE для розробки програмного забезпечення. Розробка ведеться мовами програмування C++, C#, Visual Basic .NET. Дозволяє розробляти програми с графічним інтерфейсом, а також консольні. Підтримується операційною системою Windows.

Visual Studio Code – це редактор програмного коду, який був створений Microsoft на основі Electron. Має велику бібліотеку розширень, та дозволяє писати код більш як 70 мовами програмування. Підтримується Windows та macOS, і навіть Linux. Являється досить популярним редактором коду.

NetBeans, це інтегроване середовище розробки, для різноманітних мов програмування таких як: Java, Python, PHP, JavaScript, C, C++ та Ada, однак більше орієнтоване для розробки на Java. Середовище розроблено компаніями Oracle, Apache та Sun Microsystem. Програма потребує додатково встановленого Sun JDK або J2EE SDK. Середовище розробки можна встановити на Windows, macOS, Linux та Solaris.

Eclipse – IDE розроблене компанією Eclipse. За допомоги необхідних плагінів та доповнень може розробляти програмне забезпечення на багатьох мовах, проте більше орієнтується на Java. Підтримується системами: Windows, macOS, Linux та Solaris.

Для розробки було прийнято рішення використовувати середу розробки Microsoft Visual Studio та редактор коду Visual Studio Code. IDE Visual Studio найкраще підходить для розробки програм мовою C#, а для веб розробки краще підходить VSD, за рахунок великої кількості додатків які дозволяють значно полегшити процес розробки.

3 МОДЕЛЮВАННЯ ТА РОЗРОБКА ПРОЕКТУ

3.1 Моделювання розробки проекту

Моделювання процесу розробки програмного забезпечення з використанням методології IDEF0, на рівні 0 діаграми зображено функціональний блок із усіма відповідними робочими та керуючими об'єктами. Також на діаграмі відображена уся необхідна документація та усі інструменти які використовується для створення програмного забезпечення . Діаграма нульового рівня відображена на рис. 4.

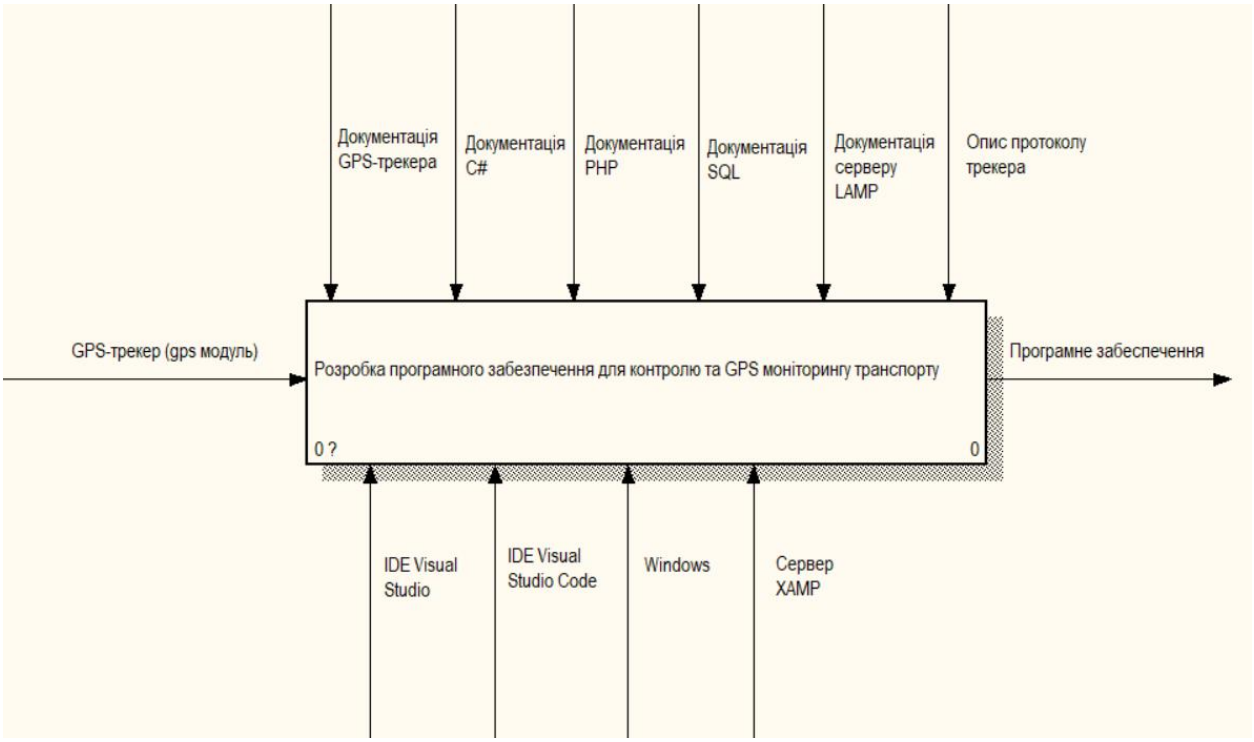


Рисунок 4. Контекстна модель. Основна діаграма.

Для реалізації проекту необхідно передбачити та відобразити послідовність подальших дій які необхідно виконати для успішного створення продукту. Ці послідовності та етапи наведені на рис. 5.

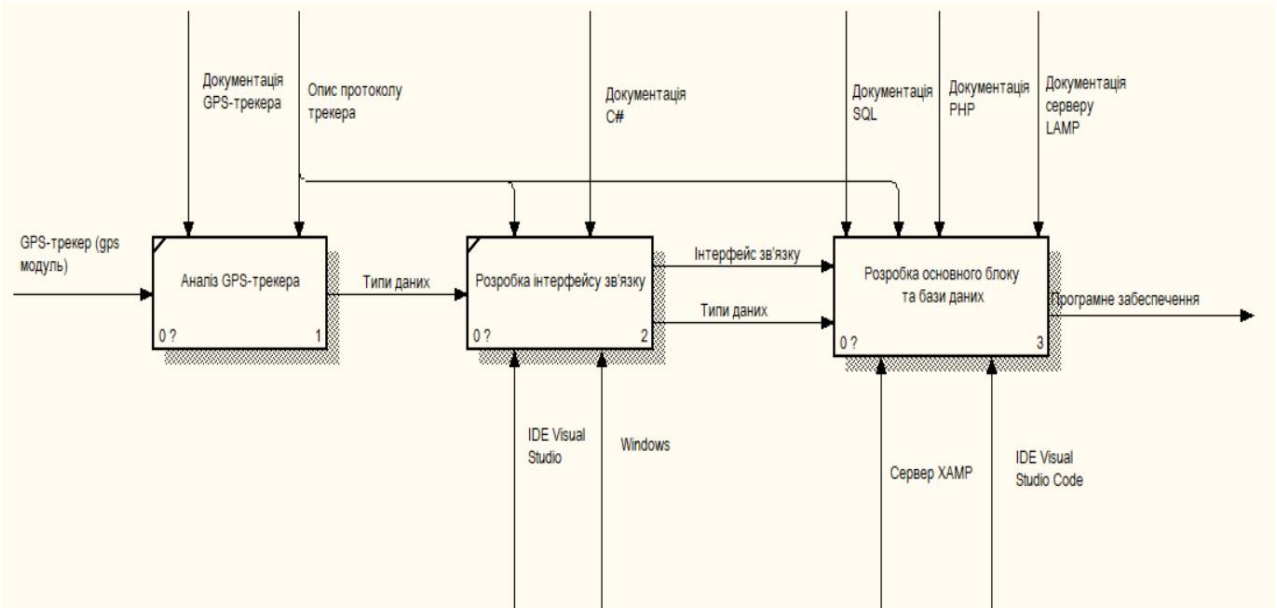


Рисунок 5. Діаграма декомпозиції. Опис структурних елементів «розробка модулю інтерфейсу зв’язку і розробка основного блоку разом із базою даних»

Для успішної реалізації запланованих робіт потрібно мати оригінальне застосування, у нашому проекті використовується трекер компанії Bitrek BI920, його документація та опис протоколу. Також необхідно мати навички налагодження веб-серверу, розуміння системи OSI і навики розробки необхідними мовами програмування, і також використання інтегрованого середовища. Результатами усього масиву робіт є створення інтерфейсу зв’язку, розробка основних алгоритмів обробки даних, створення БД та інтерфейсу користувача.

Першим кроком необхідно зробити початкові налаштування трекеру, вказати адрес сервера обробки та порт. Проаналізувати та розібратися в структурі пакетів, його протокол. Після виконання цих кроків можна вже буде виділити основні типи даних їх структуру, команди необхідні для взаємодії з пристроєм та розпланувати наступні свої кроки для подальшої розробки.

Маючи представлення як працює трекер та які пакети він відправляє, з’являється можливість розробки інтерфейсу зв’язку. Маючи інформацію про типи даних їх структуру, можна розробити базу даних та алгоритми її обробки.

Після цього розпочати розробку панель моніторингу, більш детально зображено на діаграмі композицій 3, рис. 6.

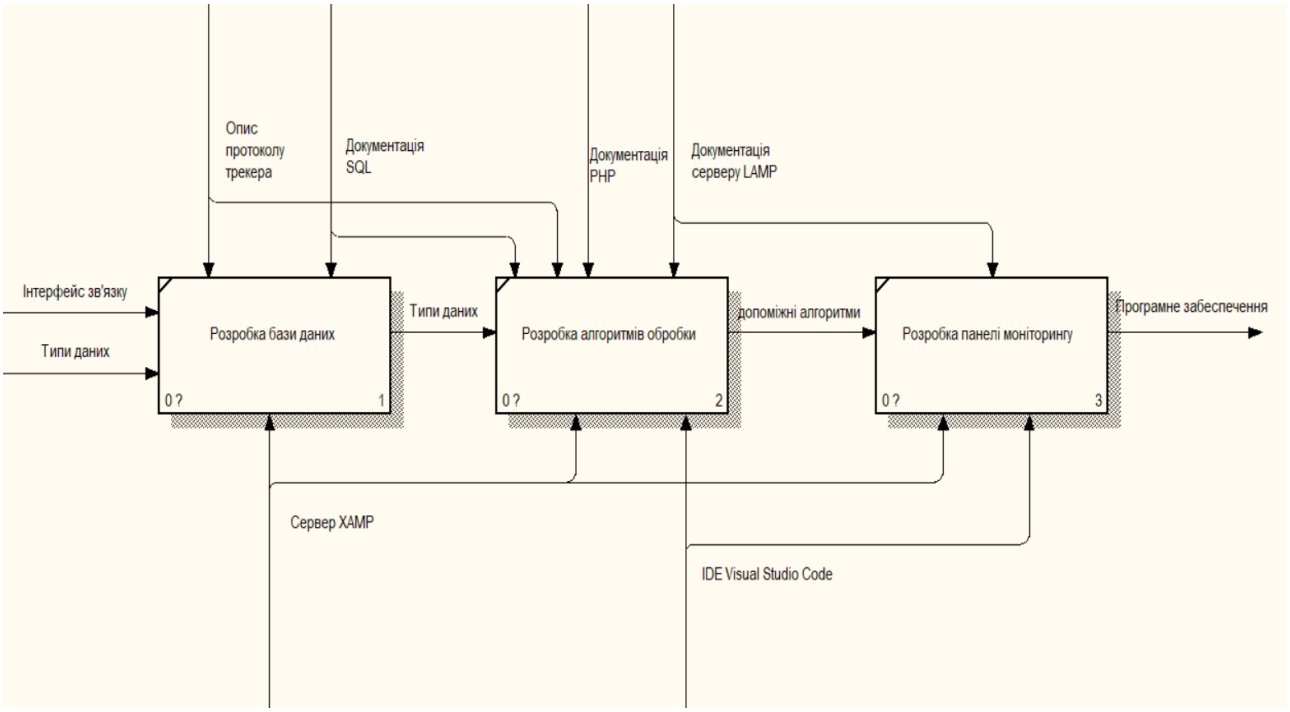


Рисунок 6. Діаграма декомпозиції 3. Опис структурного елементу основного блоку.

3.2 Дослідження GPS-трекера та його протоколу.

Для подальшої розробки програмного забезпечення необхідно розглянути принцип роботи та передачі повідомлень від gps-трекерів. Для зв’язку трекери використовують сотову мережу операторів, зокрема відправляючи повідомлення через GPRS-мережу. Зазвичай для комунікації на транспортному рівні використовуються протоколи передачі даних такі як TCP та UDP. На прикладному рівні, протоколів дуже багато, майже кожний виробник використовує власний протокол.

Дані що передається трекерами можуть бути у текстовому вигляді, так і в бінарному. Для спрощення декодування повідомлень, потокові дані розбивають на окремі відрізки. Для текстових повідомлень стандартом вважається розбиття за допомогою одного або декількох спеціальних символів. Найпоширенішим

символом, який використовується для розділення, є символ перенесення рядка з таблиці символів ASCII (код у таблиці 0x0a). У бінарних повідомленнях завжди існує заголовок який має фіксовану довжину що вимірюється у байтах, в ньому записується інформація така як довжини пакета, що вказує на розмір даних, або розмір всього повідомлення, а ще дані пристрою, наприклад його IMEI (міжнародний ідентифікатор мобільного обладнання).

Для наглядності розглянемо протокол повідомлень даних одного із gps-трекерів українського виробництва Bitrek, з яким надалі і будемо працювати. Трекери компанії Bitrek [10] є одними із найрозповсюдженішими в Україні. Для відправки даних вони використовують бінарні данні у вигляді HEX.

Протокол Bitrek [11] ми розділити на декілька етапів:

Перший етап.

Після включення живлення пристрій (gps-трекер) за замовчуванням відправляє пакет реєстраційних даних, наприклад IMEI, і далі чекає підтвердження від сервера для ініціювання сесії. Таблиця 2.

Таблиця 2, Пакет із заголовком

Довжина аних – 2 байти, бінарні дані 15 (довжина IMEI).		Ідентифікатор пристрою - IMEI. 15 байт - ASCII-кодований IMEI						
0	15	'3'	'5'	'5'	…….	'8'	'4'	'9'

Якщо сервер дозволяє з'єднання, то відправляє у відповідь 1 бінарний байт зі значенням 1. Інакше відправляє 0.

Приклад встановлення з'єднання (DEC / ASCII), :

- Трекер відправляє: 0,15,'3','5' ,'5' ,'6' ,'6' ,'7' ,'7' ,'8' ,'8' ,'9' ,'9' ,'0' ,'0' ,'1' ,'1'
- Сервер відправляє у відповідь: 1 – підтвердження з'єднання

- Сервер відправляє у відповідь: 0 - не підтвердження з'єднання (наприклад IMEI не виявлено в база даних)

Після ініціалізації з'єднання пристрій готовий до відправки даних.

Другий етап.

Після встановлення стабільного з'єднання та оновлені даних знятих за допомогою GPS, трекер починає відправляти об'єднану інформацію GPS та LBS (визначення координат за допомогою GPRS) або тільки пакет інформації GPS на сервер, із заданими інтервалами часу.

Приклад даних та його розшифровка (дані в HEX):

080100000113fc208dff00209cca800f14f650006f00d6040004000403010115031
6030001460000015d0001

Перші байти:

08 - CODEC ID. (1 байт) - константа, що визначає алгоритм розшифровки записів;

01 – 1 запис в пакеті. Кількість записів, можлива N-кількість визначає, скільки записів зі структурою.

Допоміжна інформація:

00000113fc208dff – час створення 25 Jul 2007 06:46:38 (8 байт);

00 - пріоритет 0 (1 байт).

GPS – дані (15 байт), виділені жовтим кольором:

209cca80 - довгота 547146368 = 54,7146368 º E (4 байт);

0f14f650 - широта 253032016 = 25,3032016 º N (4 байт);

006f - висота 111 метрів (2 байт);

00d6 - азимут 214 º (2 байт);

04 - 4 видимих супутника (1 байт);

0004 - швидкість 4 км / год (1 байт).

Третій етап.

Для забезпечення ефективності прийому інформації трекер після відправки даних очікує від сервера відповідь з підтверджуючими пакетами про успішне отримання інформації:

- 0 - якщо пакет має невірну контрольну суму або не розібраний;
- Число більше 0, відповідає кількісті витягнутих записів з прийнятого пакет.

Підтвердження передається у форматі 4 байт - від старшого до молодшого.

Наприклад, якщо відправлений пакет як на етапі 2, то трекер чекатиме відповідь – 00000001. У разі отримання негативного підтвердження (00000000) трекер виконає три спроби відправити дані, після чого видалить їх з пам'яті як зіпсовані.

3.3 Моделювання БД

Після дослідження пакету даних які формує gps-трекер можна перейти до проектування бази даних і таблиць, виділити примітивні типи, розробити структуру для виконання внутрішньої комунікації. Було прийнято рішення створити 4 таблиці для зберігання даних:

- таблиця tt для зберігання загальної інформації про трекер ;
- таблиця datagps для зберігання розкодованих даних трекера;
- таблиця gps_protocol для зберігання параметрів протоколу;
- таблиця gps_protocol_fields для зберігання параметрів розбору даних.

Фізична структура бази даних представлена на рис. 7:

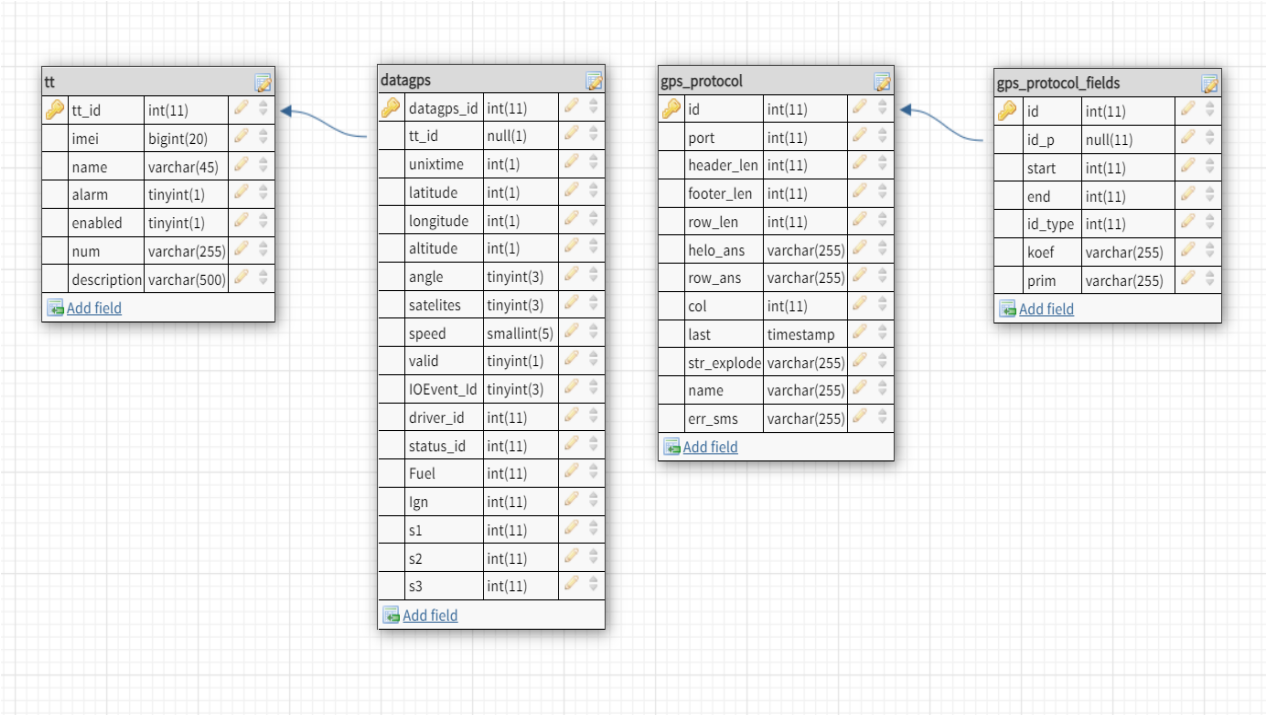


Рисунок 7. Фізична структура даних.

Таблиця tt, використовується для зберігання інформації про статичні дані трекера, такі як його imei та назву автотранспорту, або номер автомобіля. Таблиця складається із полів:

tt_id – ціле число, первинний ключ з авто-інкрементуванням, використовується як ідентифікатор;

imei – ціле число, зберігає унікальний ідентифікатор пристрою, його IMEI;

name - рядок змінної довжини, зберігає модель та назву транспортного засобу;

alarm – ціле число, зберігає дані про аварію;

enabled – ціле число, працює (1) або ні (0);

num - рядок змінної довжини, зберігає реєстраційний номер автомобілю;

description - рядок змінної довжини, зберігає додаткову інформацію про транспорт, замітки.

Команда для створення [12] таблиці:

```
CREATE TABLE `tt` (
  `tt_id` int(11) NOT NULL AUTO_INCREMENT,
  `imei` bigint(20) unsigned NOT NULL,
  `name` varchar(45) DEFAULT NULL,
  `alarm` tinyint(1) NOT NULL DEFAULT '0',
  `enabled` tinyint(1) NOT NULL,
  `num` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `description` varchar(500) DEFAULT NULL,
  PRIMARY KEY (`tt_id`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Для зберігання розкодованих даних використовується таблиця datagps, вона зберігає дані GPS та показання датчиків та має такі поля:

datagps_id – ціле число, первинний ключ з авто-інкрементуванням, використовується як ідентифікатор;

tt_id - ціле число, зберігає ID трекека;

unixtime – ціле число, зберігає дату та час у секундах, у форматі unix-time;

latitude – ціле число, зберігає географічну широту;

longitude – ціле число, зберігає географічну довготу;

altitude – ціле число, зберігається висота над рівнем моря, у метрах;

angle – ціле число, зберігає кут об’єкту (напрямок);

satelites – ціле число, зберігає кількість під’єднаних супутників;

speed – ціле число, зберігає швидкість об’єкту у км. на годину;

valid – ціле число, валідність даних (правдивість);

Fuel – ціле число, зберігає дані про залишок паливо у баку;

Ign – ціле число, зберігає інформацію про стан запалення;

s1, s2 …. s3 – цілі числа, зберігають інформацію про стан допоміжних датчиків (сенсорів) трекера, наприклад навантаження на ось, запуск або виключення двигуна. Кількість датчиків залежить від моделі, максимальна кількість 36.

Команда для створення [12] таблиці:

```
CREATE TABLE `datagps` (
  `datagps_id` int(10) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `tt_id` int(1) NOT NULL,
  `unixtime` int(1) unsigned NOT NULL,
  `latitude` int(1) NOT NULL,
  `longitude` int(1) NOT NULL,
  `altitude` int(1) unsigned NOT NULL,
  `angle` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `satelites` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `speed` smallint(5) unsigned NOT NULL DEFAULT '0',
  `valid` tinyint(1) unsigned NOT NULL DEFAULT '0',
  `Fuel` int(11) DEFAULT NULL,
  `Ign` int(11) DEFAULT NULL,
  `s1` int(11) unsigned NOT NULL DEFAULT '0',
  `s2` int(11) unsigned NOT NULL DEFAULT '0',
  `s3` int(11) unsigned NOT NULL DEFAULT '0',
  FOREIGN KEY (`tt_id`) REFERENCES `tt` (`tt_id`)
) ENGINE=InnoDB DEFAULT CHARSET= utf8;
```

Таблиця gps_protocol, розроблена для зберігання загальних налаштувань протоколів, наприклад такі як довжина заголовку, відповідь яку повинен отримати трекер для після успішного прийому даних сервером і таке інше. Таблиця має такі поля:

Id – ціле число, первинний ключ з авто-інкрементуванням, використовується як ідентифікатор;

port – ціле число, зберігає значення порту через який проходить зв’язок трекера з сервером;

header_len – ціле число, зберігає довжину заголовка;

footer_len – ціле число, зберігає довжину рядка;

row_len – ціле число, зберігає довжину закінчення;

helo_ans – рядок змінної довжини, містить дані для відповіді на вітання трекера;

row_ans – рядок змінної довжини, зберігає у собі відповідь для трекера, на прийняття даних;

col – ціле число, зберігає інформацію про кількість прийнятих пакетів;

last – календарний тип даних, зберігає інформацію про дату та час доби у вигляді кількості секунд, що пройшли з півночі 1 січня 1970 року (початок епохи UNIX);

name – рядок змінної довжини, зберігає назву протоколу;

err_sms – рядок змінної довжини, зберігає SMS відповідь для трекера у разі похибки.

Команда для створення [12] таблиці:

```
CREATE TABLE `gps_protocol` (
 `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
 `port` int(11) DEFAULT NULL,
 `header_len` int(11) DEFAULT '0',
 `footer_len` int(11) DEFAULT '0',
 `row_len` int(11) DEFAULT '0',
 `helo_ans` varchar(255) DEFAULT NULL,
 `row_ans` varchar(255) DEFAULT NULL,
 `col` int(11) DEFAULT '0',
 `last` timestamp NULL DEFAULT NULL,
 `name` varchar(255) DEFAULT NULL,
 `err_sms` varchar(255) DEFAULT NULL
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Остання таблиця gps_protocol_fields, розроблена для зберігання параметрів необхідних для розбиття пакетів та розкодування даних прийнятими від трекера і має такі поля:

Id – ціле число, первинний ключ з авто-інкрементуванням. Використовується як ідентифікатор;

id_p – ціле число, містить дані про id протоколу які узяті з таблиці gps_protocol;

start – ціле число, містить дані про початок даних (порядковий номер байту);

end – ціле число, містить дані про кінець даних (порядковий номер байту);

id_type – ціле число, зберігає інформацію про тип даних які оброблюються;

koef – рядок змінної довжини, створений для зберігання параметрів необхідного конвертування даних, наприклад для конвертування формату даних HEX у строку (=hexToStr($x));

prim – рядок змінної довжини, створений для зберігання заміток.

Команда для створення [12] таблиці:

```
CREATE TABLE `gps_protocol_fields` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `id_p` int(11) DEFAULT NULL,
  `start` int(11) DEFAULT NULL,
  `end` int(11) DEFAULT NULL,
  `id_type` int(11) DEFAULT NULL,
  `koef` varchar(255) DEFAULT NULL,
  `prim` varchar(255) DEFAULT NULL,
  FOREIGN KEY (`id_p `) REFERENCES ` gps_protocol ` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

3.4 Моделювання алгоритмів

Для успішної розробки проекту необхідно розглянути взаємодію елементів та розробити алгоритми. На рис. 8 представлена UML-діаграма [] послідовностей дій.

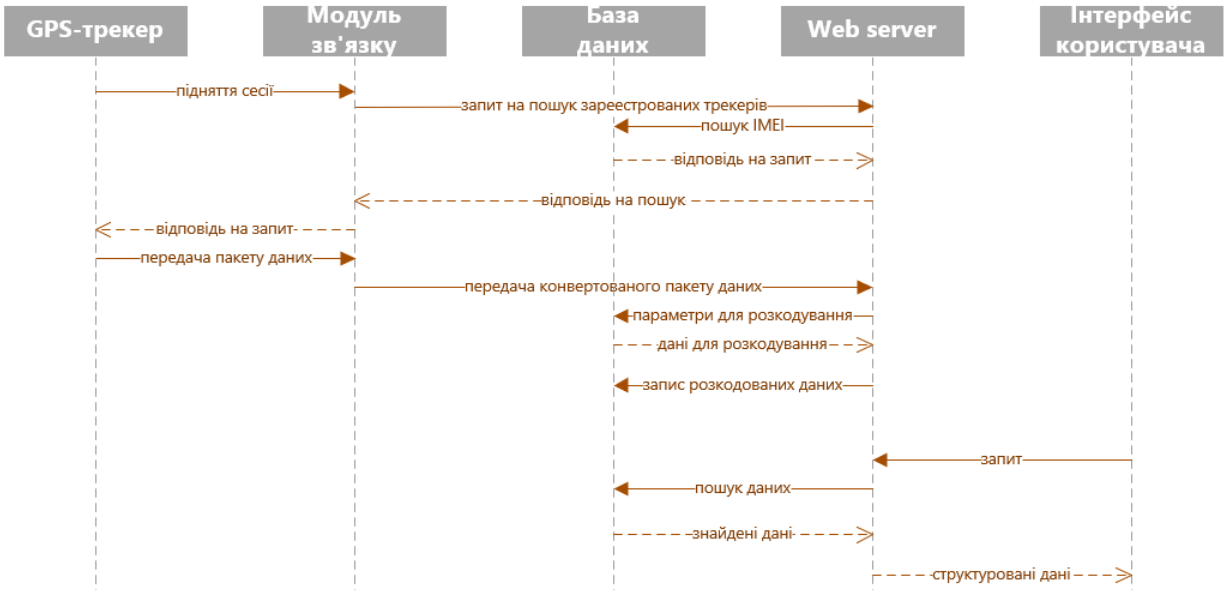


Рисунок 8. Діаграма послідовності дій у проекті

Як показано на діаграмі, для того щоб користувач почав бачити данні на інтерфейсі користувача потрібно зробити багато кроків. Розглянемо більш детально діаграму.

Почнемо з початку, після включення gps-трекера він зв’язується з сервером, та встановлює із ним сесію в нашому випадку їм виступає Модуль зв’язку, який встановлюється на спеціально виділеному для цього комп’ютері з встановленою операційною системою Windows. Після підняття сесії трекер починає передавати пакети даних. В першому пакеті міститься заголовок який містить інформація про IMEI, який потім використовується для проходження валідації. Після прийому Модуль зв’язку відокремлює IMEI та відправляє далі, для перевірки його наявності в Базі Даних. Насамперед сама перевірка проводиться модулем Веб-сервер, в якому вже прописані алгоритми для пошуку та порівняння інформації з наявними в базі. Якщо порівняння проходить успішно то йде подальший прийом пакетів даних, якщо ні то модуль зв’язку закриває сесію.

Усі інші пакети - це дані. Заголовок до розриву сесії більше не відправляється. Заголовок від даних відрізняється тільки тим, що він перший пакет в сесії і в ньому інформація про imei. В іншому це все звичайні байтові пакети.

Після прийняття пакетів Веб-сервером, він починає їх розкодовувати використовуючи для цього протокол трекера, розкодована та розсортована дані записується в базу даних. Надалі вже ця інформація використовуюся для створення звітів та відображення шляху транспорту та іншого.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Розробка модулю зв’язку.

Перед прийнятися за розробку програмного модулю котрий відповідає за розкодування пакетів даних, треба розробити блок зв’язку, який би тримав сесію так приймав пакети даних та відправляв би їх до веб серверу GET-запитами для подальшої обробки. При цьому модуль зв’язку повинен конвертувати дані в один необхідний формат.

Врахування вимоги до програмного забезпечення, при його створені було розроблено класи AVLListener, AVLServer, Program, INIManager які реалізують увесь необхідний функціонал для взаємодії з даними та сервером. На рис. 9 зображена UML-діаграма класів модулю зв’язку.

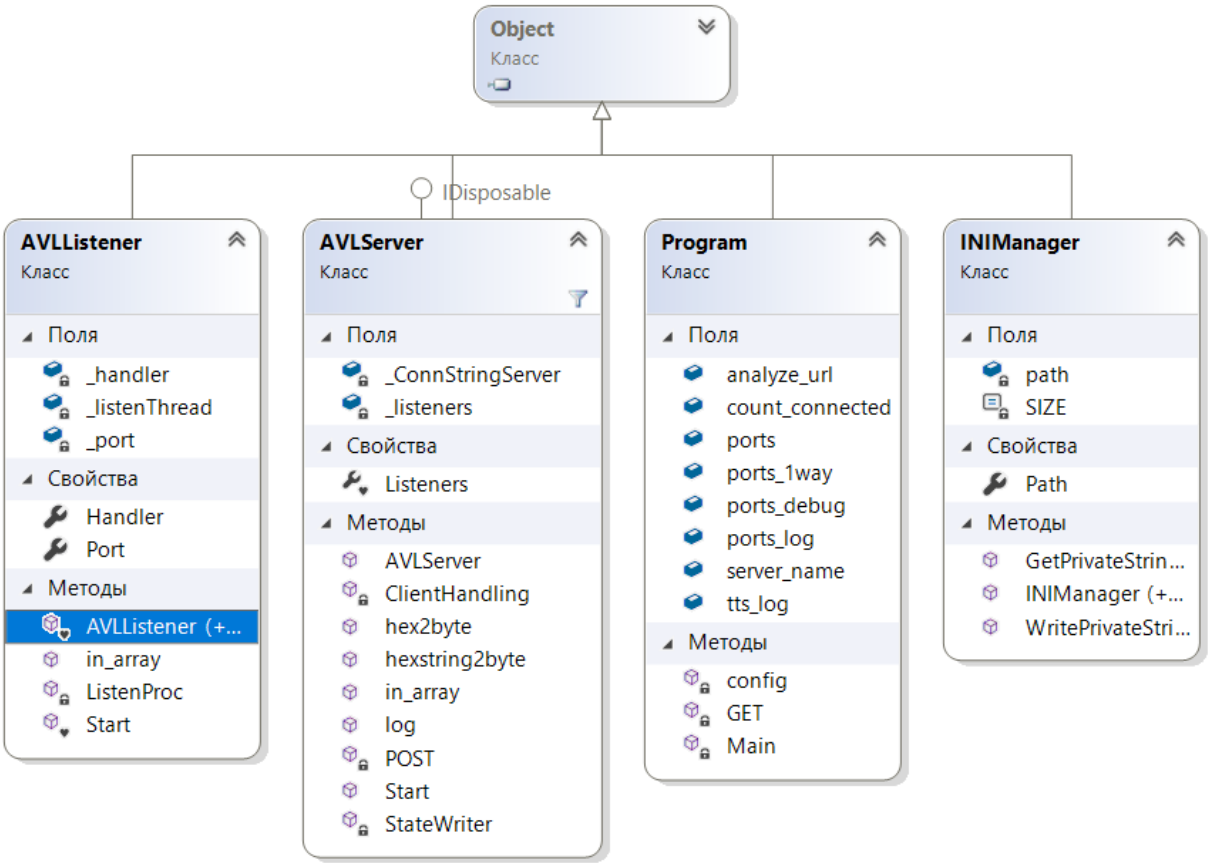


Рисунок. 9 зображена UML-діаграма класів модулю зв’язку

Клас AVLServer призначений для прийому та обробки байтів і відправлення їх далі у вигляді строки на сервер який далі буде оброблювати прийняту інформацію. Даний клас має такі поля та методи:

– Поле _listeners, це список який зберігає список портів;

– Властивість Listeners, вертає або встановлює список портів;

– Метод ClientHandling основний метод який відповідає за роботу з трекерами;

– Метод hex2byte відповідає за конвертацію HEX даних в масив байтів;

– Метод hexstring2byte, призначений для конвертування HEX рядку в масив байтів;

– Метод in_array, створений для перевірки елементів, чи являються вони членом масиву;

– Метод log призначений для виводу строки логу на екран;

– Метод POST, відправляє дані у вигляді на сервер що розкодовує дані;

– Метод Start, відповідає за запуск сервера та відкриття портів;

– Метод StateWriter створений для виводу крапок на екран, як допоміжна інформація, що програма не повисла.

Клас Program призначений для запуску програми та створення глобальних масивів які зберігають інформацію узяту з конфігураційного .ini файлу. Даний клас має такі поля та методи:

– Поле analyze_url зберігає адресу модулю який далі обробляє інформацію, яку відправляє модуль зв’язку;

– Поле count_connected зберігає інформацію про кількість підключених трекерів;

– Поле ports зберігає список портів, які потрібно відкрити для прослуховування;

– Поле ports_1way зберігає список портів, на яких працюють трекери без заголовків;

- Поле ports_debug призначене для зберігання списку портів, по котрим потрібно додатково логирувати розбір пакетів;

– Поле ports_log зберігає список портів, котрі потрібно додатково логирувати;

– Поле server_name призначене для зберігання імені серверу;

– Поле tts_log зберігає список трекерів котрі потребують додаткових логирувати;

– Метод config призначений для зчитування конфігураційного файлу config.ini;

– Метод GET створений для зчитування інформації з url ;

– Метод Main призначений для запуску програми та параметрів.

Клас AVLListener призначений для запуску портів та їх прослуховування. Даний клас має такі поля та методи:

– Метод AVLListener, створює потік;

– Метод in_array, допоміжний метод який перевіряє приналежність елементу до масиву;

– Метод ListenProc, метод який запускається якщо трекер підключається до нашого сервера;

– Метод Start, створений для запуску прослуховування портів.

Клас INIManager призначений для запуску портів та їх прослуховування. Даний клас має такі поля та методи:

– Поле path містить дані шляху до конфігураційного INI-файлу;

– Поле SIZE зберігає значення максимального розміру (для зчитування значень з файлу);

– Властивість Path, вертає або встановлює шлях до INI файлу;

– Метод GetPrivateString повертає значення з INI-файлу (по заданим секціям та ключу);

– Метод INIManager це конструктор який приймає шлях до INI-файлу;

– Метод WritePrivateString, записує значення в INI-файл (по вказаним секціям и ключу).

Для внесення параметрів за якими повинна працювати програма, було створено для зручності конфігураційній файл config.ini. Він містить у собі адресу

сервера що буде оброблювати дані, список портів які відкрити для прослуховування. Приклад конфігураційного файлу на рис. 10.

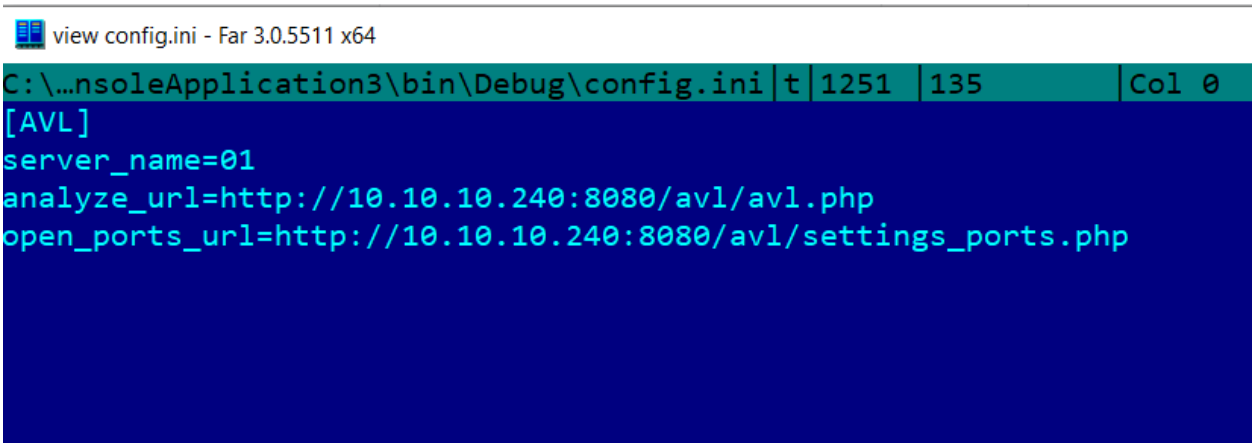


Рисунок 10. Конфігураційний файл config.ini

Далі представлено код файлу settings_ports.php, який створює масив портів які потрібно відкрити для прослуховування модулем зв’язку:

```
<?
include ('functions.php');
mysql_gps ();
$p = array();
$q = mysql_query ("select * from gps_protocol");
$n = mysql_num_rows ($q);
for ($i=0; $i<$n; $i++)
{
    $port = mysql_result ($q, $i, 'port');
    if ($port>=10000 && !in_array($port, $p))
    {
        $p[] = $port;
    }
}
$p = implode(',', $p);
print $p;
?>
```

На рис. 11 зображено роботу додатку. Спочатку ми бачимо як трекер під’єднався використовуючи для спілкування порт 44005, далі він відправив пакет з привітанням, та пакет з даними. Після цього додаток відправляє дані до серверу. Розміри пакетів відображаються у байтах.

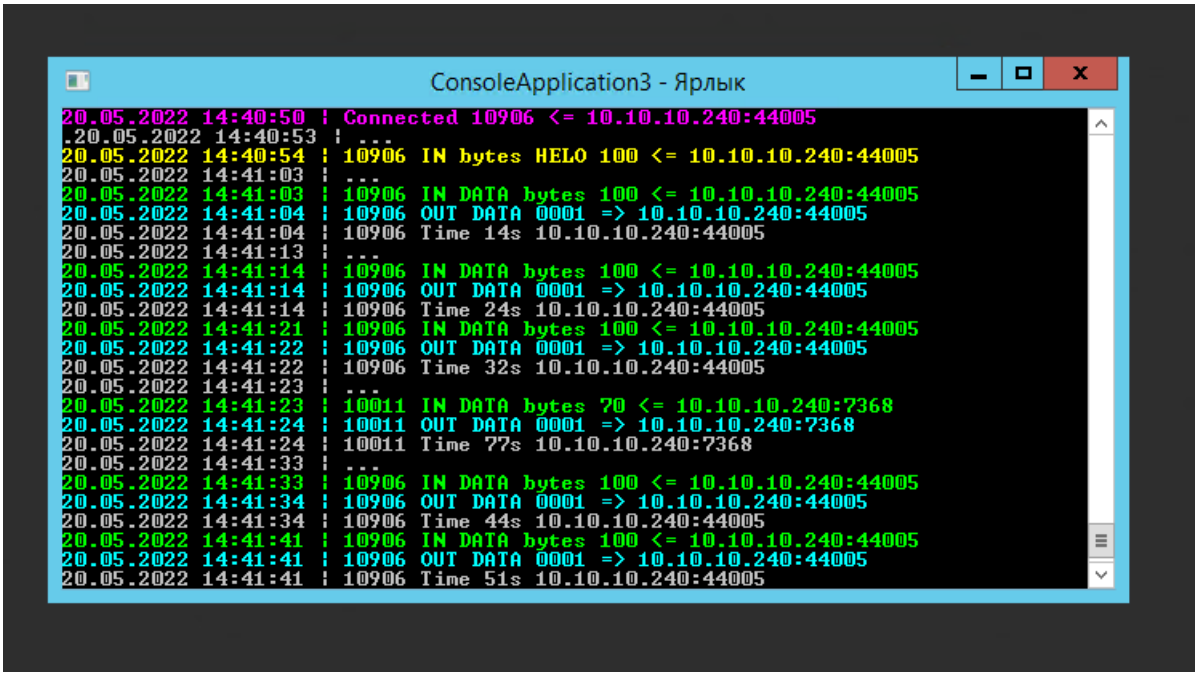


Рисунок 11. Приклад роботи додатку який спілкується з трекером.

4.2 Розробка модулю розкодування даних та настройки протоколу.

Після вивчення документації протоколів за якими працюють трекери, в розділі Моделювання та розробка БД було створено дві таблиці для роботи з протоколами. Таблиці було створено за допомоги Bootstrap. В одній зберігаються параметри протоколу [13,14] таблиця gps_protocol, а в іншій таблиці gps_protocol_fields інформація про те де і в яких байтах записана інформація. Для зручності роботи з цими таблицями було розроблено графічну панель, через яку можливо вносити потрібну інформацію про різні параметри протоколу які були раніше розглянуті. На рис. 12 зображено панель в яку вносяться параметри протоколу.

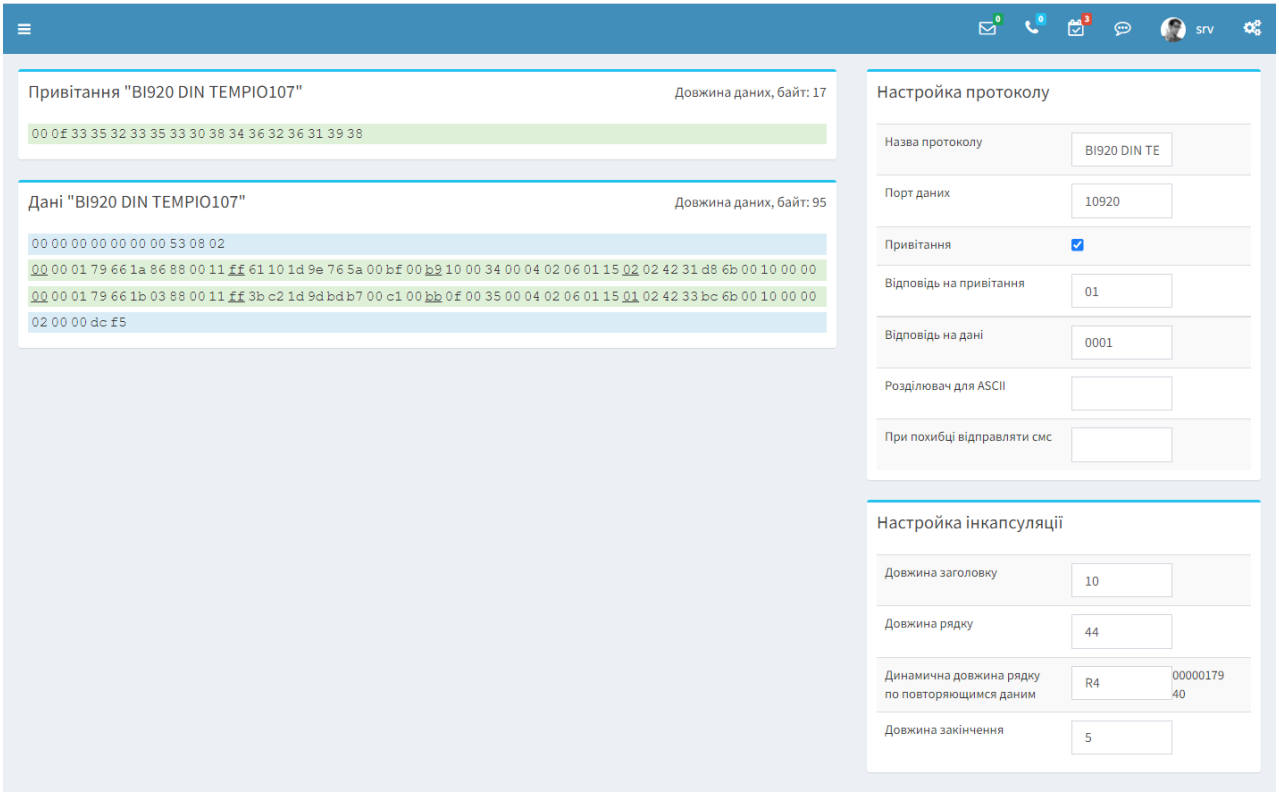


Рисунок 12. Панель яка відповідальна за параметри протоколу.

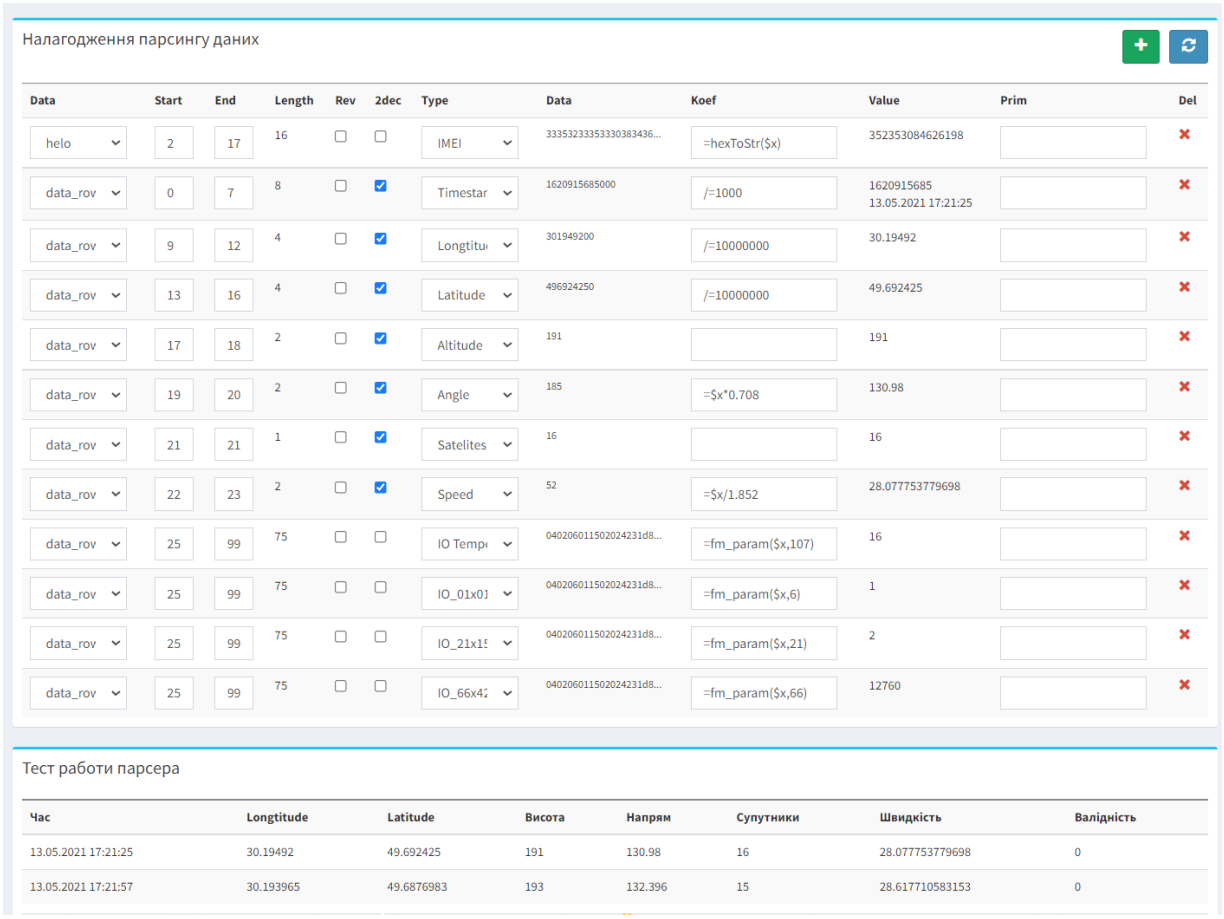


Рисунок 13. Панель яка відповідальна за параметри розбору протоколу.

Для роботи з таблицею gps_protocol_fields була розроблена допоміжна панель. Вона дозволяє вносити дані про розміри частин на які потрібно розділити пакет для його розкодування, та інформацію за що вони відповідають. Також вона відображає розкодовані дані. Панель зображена на рис.13.

Панель має такі поля:

– Data, відповідає за тип пакету, наприклад, пакет привітання (helo) або пакет даних (data_row);

– Start та End, кінець та початок, порядковий номер байту;

– Length, інформаційний стовбець, відображає довжину;

– Rev, параметр який відповідає за зчитування байту у реверсному порядку (праворуч на ліво);

– 2dec, параметр який відповідає за конвертування hex у dec;

– Type, відповідає за вибір типу даних які розкодовуються такі як IMEI, довгота, широта, час, висота, напрям та ін.;

– Data, інформаційний стовбець, відображає інформацію що була отримана після розкодування;

– Koef

– Value, інформаційний стовбець, відповідає за відображення кінцевого результату;

– Prim поле створене для заміток.

Після внесення параметрів та збереження їх у базі даних, починає працювати алгоритм розбору пакету та розкодування даних, які потім записуються в окрему таблицю datagps.

На мал, 14 представлена панель яка відображає маршрут транспортного засобу, для цього було використано мапу Google maps, а для прорисовки лінії пройденого шляху та точок було використано функцію JS “google.maps.Polyline” [16] компанії Google.

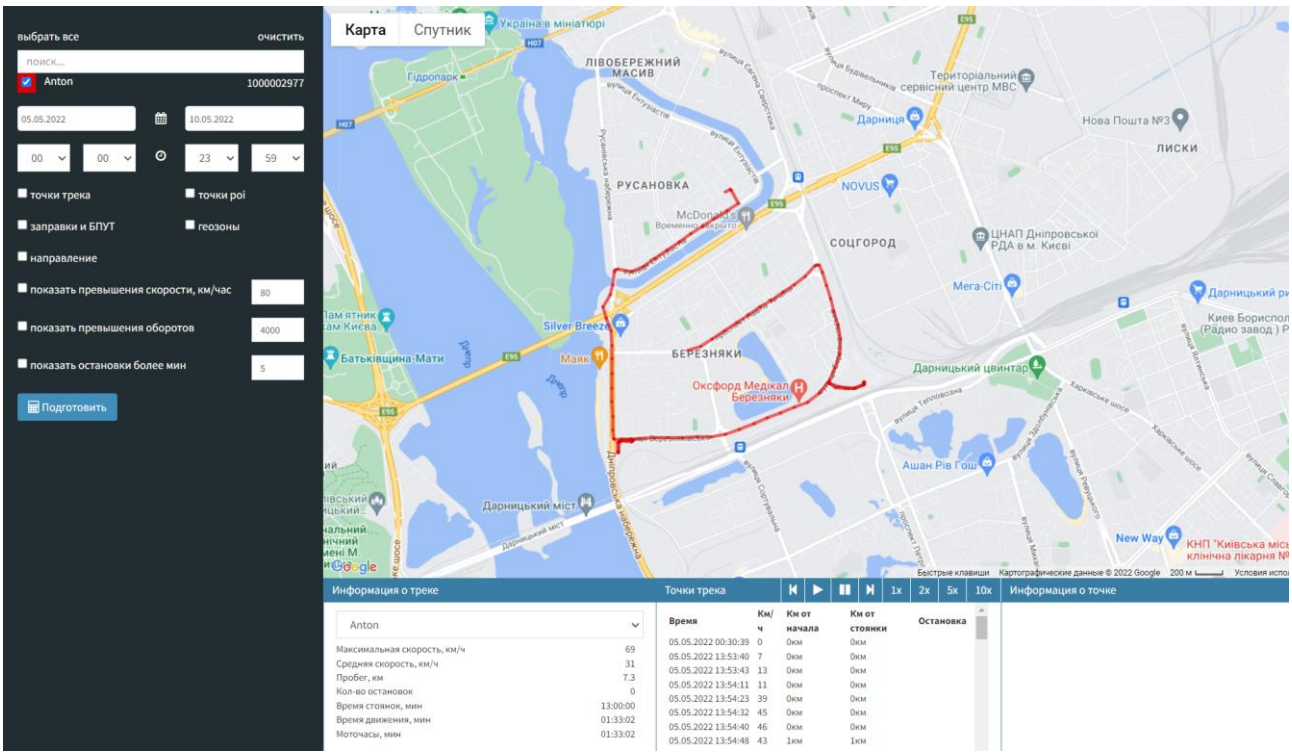


Рисунок 14. Панель відображення пройденого маршруту.

На рис. 15 та 16 зображено інформацію про стан автомобіля, його положення, швидкість, кількість палива.

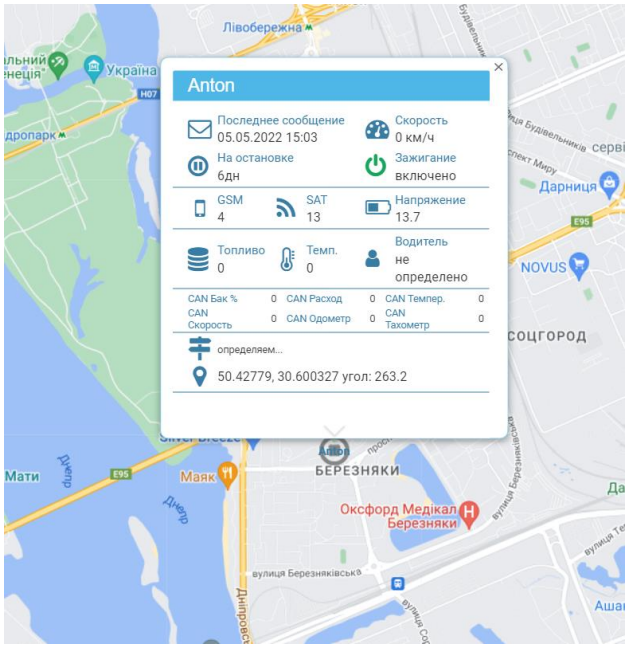


Рисунок 15. Інформація про автомобіль.

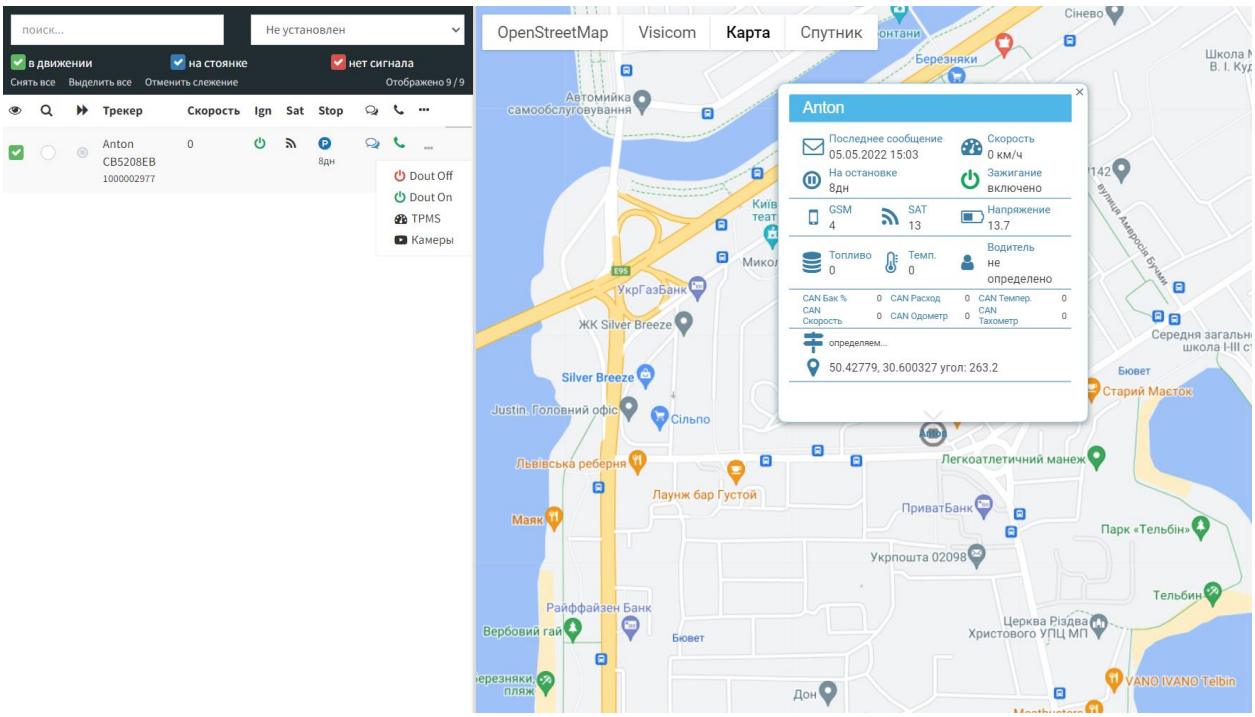


Рисунок 16. Інформаційна панель.

ВИСНОВКИ

Під час реалізації проекту були проведено предметний аналіз області, розглянуті технології супутникового моніторингу, спроектовано та реалізовано програмній продукт який складається з двох частин: блоку зв’язку та модулю обробки інформації. Для розробки були використані сучасні технології та програмні додадки необхідні для створення та розробки програмного продукту. Було розглянуто та досліджено протокол gps-трекера який використовується для передачі даних. При детальному аналізі протоколу було виявлено головні елементи, необхідні для розробки структури бази даних, та спроектовано саму БД.

Був реалізовано програмний модуль який дозволяє приймати пакети даних, піднімати сесії, конвертувати та передавати дані далі. Цей модуль будо вирушено створити як окрему програму що запускається окремо на виділеному для цього сервері або ПК.

Інший модуль було створено для встановлення на веб-сервері, що дозволить легко корегувати код, розширювати функціонал новими додатками, без переустановлення програми. Була реалізована можливість створення конфігурацій розбору для різних пакетів роботи трекерів, на окремому інтерфейсі. Та створено панель gps-моніторингу за транспортом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. GPS [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Global_Positioning_System

2. GPS-трекер [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/GPS_tracking_unit.

3. Клієнт-серверна_архітектура – Вікіпедія. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура.

4. Триярусна_архітектура – Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Триярусна_архітектура .

5. Самі популярні субд: рейтинг 2018-го року. [Електронний ресурс] – Режим доступу до ресурсу: https://itsource.com.ua/blog/samye-populjarnye-subd-rejting-2018-go-goda/

6. Microsoft SQL Server – Вікіпедія. [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Microsoft_SQL_Server (дата звернення 09.02.2020).

7. Що таке PHP? [Електронний ресурс] – Режим доступу до ресурсу: https://www.php.net/manual/en/intro-whatis.php. (дата звернення: 22.04.2020).

8. Кращі мови програмування 2020 року, які варто вивчати [Електронний ресурс] – Режим доступу до ресурсу: https://merehead.com/ru/blog/popular-programming-languages-2020/. (дата звернення: 22.04.2020).

9. Діаграми UML [Електронний ресурс] – Режим доступу до ресурсу: https://planerka.info/item/diagrammy-kommunikacij-uml/. (дата звернення: 01.05.2020).

10. Опис протоколу BITREK – Режим доступу до ресурсу: https://gpsm.ua/downloads/Bitrek/ProtokolBitrek/BITREK_protocol_RU.pdf.

11. Пристрій спостереження за рухомими об'єктами BI 910 TREK – Режим доступу до ресурсу: https://gpsm.ua/downloads/Bitrek/910/BI%20910%20TREK_manual_RUS_v.2019.09.1.pdf.

12. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL: повнеруководство, 3-е виданя SQL: The Complete Reference, Third Edition. М.: «Вильямс», 2014,960 с. ISBN 978-5-8459-1654-9.

13. Меєр Е., «CSS. Карманный справочник». –Диалектика, – 2020. – 208 с. 20.

14. Роббинс Д., «HTML 5. Карманный справочник». –Диалектика, – 2020. – 192 с. 21.

15. Фленаган Д., «JavaScript. Карманный справочник». –Диалектика, – 2020. – 320 с.

16. Simple Polylines [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/maps/documentation/javascript/examples/polyline-simple.

17. Добрівський О.В., Гаманюк І.М. Моніторинг сільського господарства за допомогою gps-технологій, Науково-технічна конференція «Застосування програмного забезпечення в ІКТ», – 2022. – 66 с. [Електронний ресурс] – Режим доступу до ресурсу: https://dut.edu.ua/uploads/p_2121_99261151.pdf.

18. Добрівський О.В., Гаманюк І.М. Енергоефективність модулів gps за допомогою розрахунків в хмарі, Науково-технічна конференція «Застосування програмного забезпечення в ІКТ», – 2022. – 84 с. [Електронний ресурс] – Режим доступу до ресурсу: https://dut.edu.ua/uploads/p_2121_99261151.pdf

Додаток А

Лістинг коду класу AVLListener, який відповідає за відкриття портів для прослуховування.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Sockets;
using System.Net;
using System.Threading;

namespace MGPS.AVL
{
    class AVLListener
    {
        private int _port = 8888;
        private ParameterizedThreadStart _handler;
        private Thread _listenThread;
        internal AVLListener(int port, ParameterizedThreadStart handler)
        {
            _port = port;
            _handler = handler;
        }
        internal AVLListener(int port)
        {
            _port = port;
        }
        public int Port
        {
            get { return _port; }
            //set { _port = value; }
        }
        public static bool in_array(String[] arr, string need)
        {
            bool f = false;
            for (int i = 0; i < arr.Length; i++)
            {
                if (arr[i] == need) f = true;
            }
```

```
            return f;
        }
        public ParameterizedThreadStart Handler
        {
            get { return _handler; }
            set { _handler = value; }
        }
        internal void Start()
        {
            TcpListener parameter = new TcpListener(IPAddress.Any, this._port);
            _listenThread = new Thread(new ParameterizedThreadStart(this.ListenProc));
            _listenThread.Start(parameter);
        }
        private void ListenProc(object listener)
        {
            TcpListener listener2 = (TcpListener)listener;
            listener2.Start();

            while (true)
            {
                //var colx = listener2.
                Console.WriteLine("Connected = " +
MYAVL.Program.count_connected.ToString());

                Thread th = new Thread(new ParameterizedThreadStart(this._handler));
                try
                {
                    TcpClient parameter = listener2.AcceptTcpClient();

                    parameter.SendTimeout = 300000;
                    parameter.ReceiveTimeout = 300000;

                    Console.ForegroundColor = ConsoleColor.Magenta;
                    if (in_array(MYAVL.Program.ports_log, _port.ToString()))
                        Console.WriteLine(DateTime.Now + " | " + _port + " | Connected
<= " + parameter.Client.RemoteEndPoint);
                    Console.ForegroundColor = ConsoleColor.Gray;

                    MYAVL.Program.count_connected++;
                    MYAVL.Work tt = new MYAVL.Work();

                    th.Start(tt.ProcessConnection(parameter));
                    Thread.Sleep(75);
```

```
                }
                catch (Exception exception)
                {
                    Console.WriteLine("Some Exception in AVLListener ");

                    try
                    {
                        th.Interrupt();
                        th = null;
                    }
                    catch { }
                    }
            }
        }
    }
}
```

Далі представлено код методу POST класу AVLServer, який відправляє дані методом POST:

```
private static string POST(string Url, string Data) // відправка даних post
{
    System.Net.WebRequest req = System.Net.WebRequest.Create(Url);
    req.Method = "POST";
    req.Timeout = 10000;
    req.Proxy = null;
    req.ContentType = "application/x-www-form-urlencoded";
    byte[] sentData = Encoding.UTF8.GetBytes(Data);
    req.ContentLength = sentData.Length;

    using (var stream = req.GetRequestStream())
    {
        stream.Write(sentData, 0, sentData.Length);
    }

    var response = (HttpWebResponse)req.GetResponse();
    var Out = new StreamReader(response.GetResponseStream()).ReadToEnd();
    return Out;
 }
```

Далі представлено частину коду класу AVLServer яка створена для перетворення HEX строку в масив байтів:

```
public static byte[] hexstring2byte(string hexString)
{
    if (hexString.Length % 2 != 0)
    {
        throw new ArgumentException(String.Format(CultureInfo.InvariantCulture, "The
binary key cannot have an odd number of digits: {0}", hexString));
    }
    byte[] data = new byte[hexString.Length / 2];
    for (int index = 0; index < data.Length; index++)
    {
        string byteValue = hexString.Substring(index * 2, 2);
        data[index] = byte.Parse(byteValue, NumberStyles.HexNumber,
CultureInfo.InvariantCulture);
    }
    return data;
}
```

Далі представлено частину коду класу AVLServer яка створена для перетворення HEX в масив байтів:

```
public static byte[] hex2byte (string str)  // репеводит HEX  в массив байтов
{
    List<byte> listHeader = new List<byte>();
    char[] charHeader = str.ToCharArray();
    foreach (char x in charHeader)
    {
        byte val = (byte)char.GetNumericValue(x);
        listHeader.Add(val);
    }
    byte[] byteHeder = listHeader.ToArray();
    return byteHeder;
}
```

Далі представлено код методу GET класу Program , який дозволяє отримувати дані в строку з url адрес, та прийняті данні :

```
    private static string GET(string Url)
    {
        System.Net.WebRequest req = System.Net.WebRequest.Create(Url);
        req.Method = "POST";
        req.Timeout = 100000;
        req.ContentType = "application/x-www-form-urlencoded";

        System.IO.Stream sendStream = req.GetRequestStream();
        Thread.Sleep(500);

        sendStream.Close();
        System.Net.WebResponse res = req.GetResponse();
        System.IO.Stream ReceiveStream = res.GetResponseStream();
        System.IO.StreamReader sr = new System.IO.StreamReader(ReceiveStream,
Encoding.GetEncoding("UTF-8"));
        Char[] read = new Char[256];
        int count = sr.Read(read, 0, 256);
        string Out = String.Empty;
        while (count > 0)
        {
            String str = new String(read, 0, count);
            Out += str;
            count = sr.Read(read, 0, 256);
        }
        return Out;
    }
```

Приклад коду збереження даних в бд, в даному випадку збереження розродованих даних у базу datagps:

```
    mysql_query("insert into $dbname.log_io set tt_id='$tt_id',
unixtime='".$x['unixtime']."', io_id=$io_id, sensor_id='8', val='$ign', datagps_id='$idi',
latitude='$lat', longitude='$lon'", $con1);
```

Далі представлено код який використовується для відображення пройденого шляху по даним з трекера. Скрипт за допомоги AJAX зв’язується з розробленим API get_track.php котрий вертає масив з координат:

```
    var point = new google.maps.LatLng(responce.track[i].lat, responce.track[i].lon);

    if (i>0) {
        var flightPlanCoordinates = [];
        var point = new google.maps.LatLng(responce.track[i-1].lat, responce.track[i-
1].lon);
        flightPlanCoordinates.push(point);
        var point = new google.maps.LatLng(responce.track[i].lat, responce.track[i].lon);
        flightPlanCoordinates.push(point);

        if ($('#is_arrow').attr("checked") == 'checked') {
            var flightPath = new google.maps.Polyline({
                path: flightPlanCoordinates,
                icons: [{
                    icon: {path: google.maps.SymbolPath.FORWARD_CLOSED_ARROW},
                    offset: '100%'
                }],
                geodesic: true,
                strokeColor: color,
                strokeOpacity: 0.6,
                strokeWeight: 4
            });
        } else
        {
            var flightPath = new google.maps.Polyline({
                path: flightPlanCoordinates,
                geodesic: true,
                strokeColor: color,
                strokeOpacity: 0.6,
                strokeWeight: 4
            });
        }
        flightPath.setMap(map);
        google.maps.event.addListener(flightPath, 'click', function() {
            console.log(tt_id);
            $('.tr_tts').css("background-color","");
            $('.tr_tts_'+tt_id).css("background-color","#357CA5");
            $('#ttsel').val(tt_id);
            tt_sel();
        });
    }
```

Далі представлен код API get_track.php який створений для створення масиву точок з координат за заданий період часу, а також інші данні як швидкість та дані деяких датчиків.

```
<?
ini_set('date.timezone', 'Europe/Kiev');
date_default_timezone_set ('Europe/Kiev');

chdir ('../');
include "config.inc";

include ('includes/functions.php');
mysql_gps();

$tt_id = $_REQUEST['tt_id'];
$t1 = $_REQUEST['t1'];
$t2 = $_REQUEST['t2'];


$limit = $_REQUEST['limit'];

if ($limit>2000000) $limit=2000000;
if (!$limit) $limit=2000000;

$q = mysql_query ("select * from tt where tt_id='$tt_id'");
$id_company = mysql_result ($q, 0, 'company_id');

$q = mysql_query ("select * from company where company_id='$id_company'");
$dbname = mysql_result ($q, 0, 'dbname');

$where = '';

if ($t1) $where .= " and unixtime>='$t1' ";
if ($t2) $where .= " and unixtime<='$t2' ";

$q = mysql_query ("select
unixtime,latitude,longitude,speed,valid,s1,s2,s5,s9,s24,s25,s26,s27,s28,s29,s36 from
$dbname.datagps where tt_id='$tt_id' $where order by unixtime limit $limit");
$n = mysql_num_rows ($q);

$a = array();
$dist = 0;
```

```
$dist2 = 0;

$i=-1;

while ($row = mysql_fetch_array($q, MYSQL_ASSOC))
{
 $valid = $row['valid'];

 if ($valid) {
  $i++;
  $t = date('d.m.Y H:i:s', $row['unixtime']);
  $x = $row['latitude'] / 600000;
  $y = $row['longitude'] / 600000;
  $speed = $row['speed'] * 1.857;
  $valid = $row['valid'];
  $ut = $row['unixtime'];


  if ($valid && $x && $y) {
   $z['t'] = $row['unixtime'];
   $z['lat'] = round($x, 6);
   $z['lon'] = round($y, 6);
   $z['s'] = round($speed);
   $z['i'] = $row['s1']||$row['s2']?1:0;
   $z['p'] = round($row['s5']);
   $z['c1'] = round($row['s24']);
   $z['c2'] = round($row['s25']);
   $z['c3'] = round($row['s26']);
   $z['c4'] = round($row['s27']);
   $z['c5'] = round($row['s28']);
   $z['c6'] = round($row['s29']);

   if ($i) // calc dist
    {
     $ddist = calculateTheDistance($x_old, $y_old, $x, $y);

     if ($ddist < 50000) {
      $dist += $ddist;
      $dist2 += $ddist;
      }
    }
    $x_old = $x;
    $y_old = $y;
```

```
    $last_unixtime = $row['unixtime'];

    $z['dx'] = round($dist);
            $z['dx2'] = round($dist2);
            $a[] = $z;
  }
 }
}

print json_encode($a);
?>
```

Далі приведено код функції function calculateTheDistance яка створена для розрахунку відстані по координатам:

```
function calculateTheDistance ($fA, $lA, $fB, $lB) {
     // переведення координат в радіани
    $lat1 = $fA * M_PI / 180;
    $lat2 = $fB * M_PI / 180;
    $long1 = $lA * M_PI / 180;
    $long2 = $lB * M_PI / 180;
   // косинуси и синуси широт и різниця довготи
    $cl1 = cos($lat1);
    $cl2 = cos($lat2);
    $sl1 = sin($lat1);
    $sl2 = sin($lat2);
    $delta = $long2 - $long1;
    $cdelta = cos($delta);
    $sdelta = sin($delta);
    // обчислення довжини великого кола
    $y = sqrt(pow($cl2 * $sdelta, 2) + pow($cl1 * $sl2 - $sl1 * $cl2 * $cdelta, 2));
    $x = $sl1 * $sl2 + $cl1 * $cl2 * $cdelta;
     //
    $ad = atan2($y, $x);
    $dist = $ad * 6372795;

    return $dist;
}
```

Далі представлено код API файлу settings_ports.php, який створений для створення масиву портів які потрібно відкрити для прослуховування модулем зв’язку:

```
<?
include ('functions.php');
mysql_gps ();
$p = array();
$q = mysql_query ("select * from gps_protocol");
$n = mysql_num_rows ($q);
for ($i=0; $i<$n; $i++)
{
 $port = mysql_result ($q, $i, 'port');
 if ($port>=10000 && !in_array($port, $p))
 {
  $p[] = $port;
 }
}
$p = implode(',', $p);
print $p;
?>
```

Далі представлена функція function get_arr яка створеня для роздулу масиву на задані частини, та виводу їх в окремих контейнерах. Данна функція використовується для розділення даних трекетера на окремі відрузки які потім декодуються:

```
function get_arr ($a, $start, $end, $fclear=0, $is_rev=0)
{
 $out = '';
 if (is_numeric($start) && is_numeric($end) && $end>=$start)
  {
   if ($is_rev)
   {
    for ($i=$end; $i>=$start; $i--)
     {
      $out .= $fclear?$a[$i]:'<span class="gh">'.$a[$i].'</span>';
     }
```

```
   }else
    {
     for ($i=$start; $i<=$end; $i++)
     {
      @$out .= $fclear?$a[$i]:'<span class="gh">'.$a[$i].'</span>';
     }
    }
  }
   return $out;
}
```

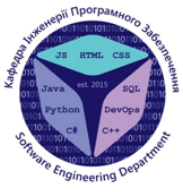
Додаток Б

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ



РОЗРОБКА СИСТЕМИ ДЛЯ ЗАБЕЗПЕЧЕННЯ КОНТРОЛЮ ТРАНСПОРТНИХ ЗАСОБІВ МОВОЮ C#

Виконав студент 4 курсу
Групи ПД-42
Добрівський О.В.
Керівник роботи
Трінтіна Н. А.

Київ -2022

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- Мета роботи: розробка програмного засобу, який дозволяє проводити моніторинг транспортних засобів, відображати місцезнаходження об’єкту, пройдений шлях.
- Об’єкт дослідження: процес розробки програмного забезпечення яке може здійснювати контроль за транспортними засобами.
- Предмет дослідження: система забезпечення контролю транспортними засобами.

АКТУАЛЬНІСТЬ

В наш час системи контролю можуть мати велике значення у випадку перевезення цінних вантажів або при перевезеннях які потребують особливого контролю з інших причин (наприклад, стратегічних). У зв'язку з цим до класу користувачів можуть відноситися митні служби, поліція, армія, прикордонна служба і навіть охоронні підприємства, що спеціалізуються наприклад на перевезенні грошей, спеціальних вантажів (радіоактивних матеріалів), палива, та інше.

За рахунок збільшення автомобільного транспорту як засобу сполучення, що використовується як для перевезення людей, так і вантажів різного типу, складу і призначення. Збільшується і попит на системи контролю транспортних засобів, які би дозволяли проводити технічний та логістичний моніторинг транспорту, а також моніторинг безпеки.

3

Перелік загальних можливостей контролю транспорту.

1. відстежувати маршрут руху, швидкість у будь-якій точці, місця та тривалість стоянок, витрата палива;
2. виявляти несанкціоновані маршрути та дії водія;
3. відображати маршрут та розташування об'єктів на електронній карті як в Україні, так і в Європі;
4. оптимізувати витрату палива та часу за допомогою постійного контролю за дотриманням маршруту;
5. безпосередньо контролювати безпеку вантажів, отримуючи миттєві оповіщення про будь-які несанкціоновані спроби розкриття вантажних відсіків, керувати замикаючими пристроями вантажного відсіку (опціонально);
6. дистанційно керувати бортовими пристроями автомобіля (дистанційний іммобілайзер);

4

Зведені результати аналізу характеристик аналогів ігор в жанрі візуальної ноели

Параметри	2Control	Wialon	Verizon
Типи трекерів	124	107	78
К-сть протоколів	104	86	63
Підтримка моніторингу тиску у шинах	Так	Ні	Так
К-сть дадаткових цифрових та аналогових входів	64	32	24
Візуалізація hex трафику	Так	Ні	Ні
Графічне налагодження нових протоколів	Так	Ні	Ні
Логування роботи протоколів	Так	Так	Ні
Робота із ASCI протоколом	Так	Так	Ні
Робота по базовим станціям без GPS	Так	Так	Ні
Підтримка цифрових виходів	8	4	2

5

ТЕХНІЧНІ ЗАВДАННЯ

- Спроектувати архітектуру програми;
- По можливості реалізувати програму таким чином, щоб її можна було легко оновлювати при необхідності;
- Додаток повинен бути легко масштабуватися і доповнюватися;
- Архітектура програми має бути зрозумілою, логічною та обґрунтованою;
- Додаток має бути кросплатформеним;
- Реалізувати веб-інтерфейс програми;
- Реалізувати програму у форматі трирівневої архітектури.

6

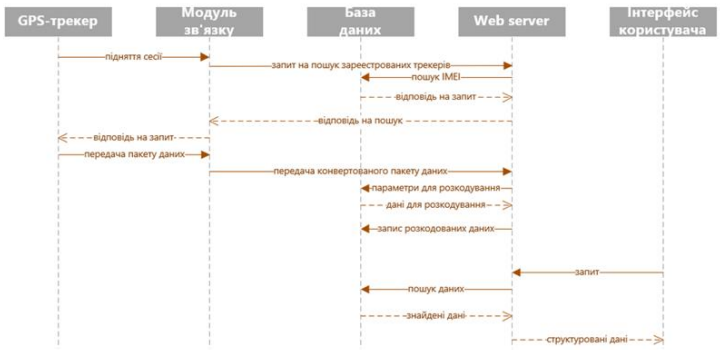
Програмні засоби та інструменти реалізації

Проаналізовано та обрано для реалізації такі інструменти:

- Середу розробки Microsoft Visual Studio для розробки додатку для Windows;
- Microsoft Visual Studio Code для розробки веб-додатку, та XAMP;

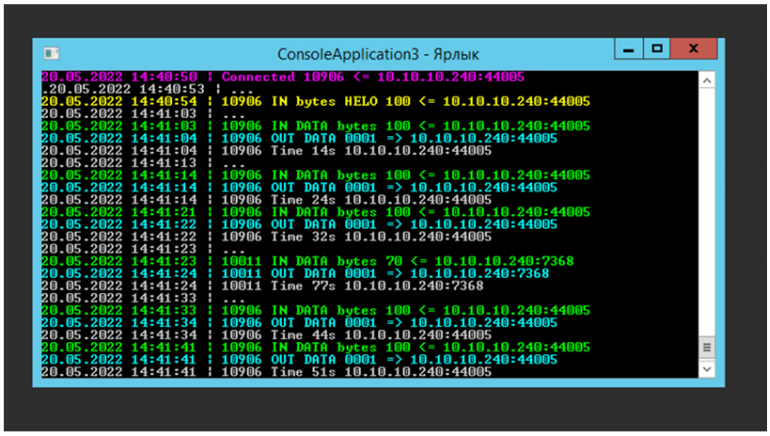
7

Діаграма послідовності дій

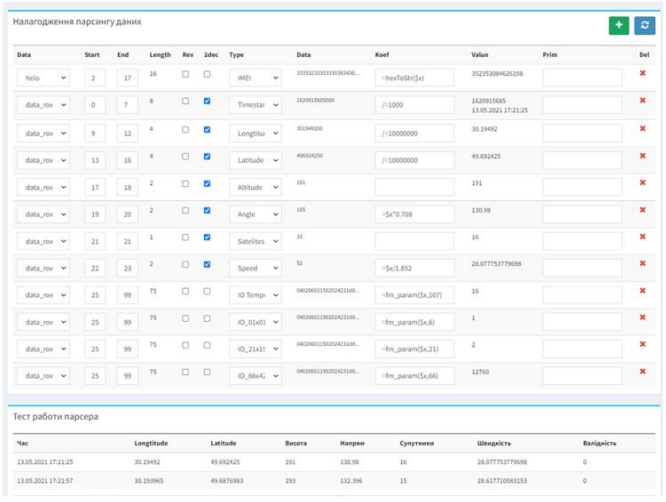


8

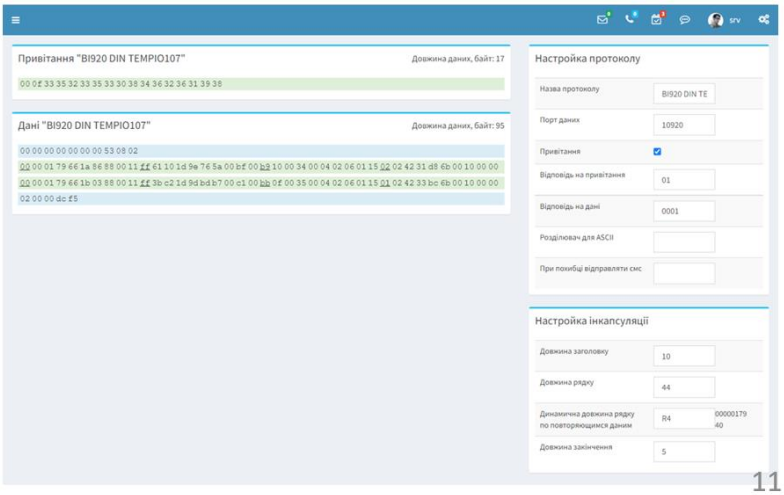
Приклад роботи програми що використовується для зв’язку з трекереми



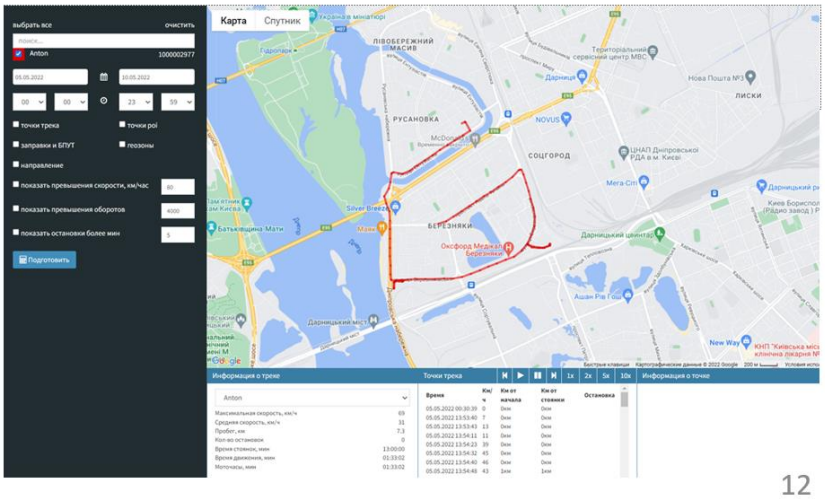
9



10



11



12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛУДЖЕННЯ

1. Добрівський О.В., Гаманюк І.М. Моніторинг сільського господарства за допомогою gps-технологій, Науково-технічна конференція «Застосування програмного забезпечення в ІКТ», – 2022, Напрям Автоматизація та управління бізнес процесами.

2. Добрівський О.В., Гаманюк І.М. Енергоефективність модулів gps за допомогою розрахунків в хмарі, Науково-технічна конференція «Застосування програмного забезпечення в ІКТ», – 2022, Напрям Хмарні Технології.

13

ВИСНОВКИ

1. Проведено аналіз розробки програмного забезпечення для контролю транспортних засобів.
2. Розглянуті технології для моніторингу транспорту.
3. Визначено головні функції для майбутнього додатку, та технічні засоби для їх реалізації.
4. Розроблено додаток для зв'язку з трекерами.
5. Алгоритм розбору пакетів даних.

Перспективи подальших досліджень та розвитку роботи:

1. Адаптація додатку під мобільні пристрої;
2. Додання додаткових функцій для контролю пального, візуалізації триангуляції, інформування про ДТП та ін.;
3. Розробити можливість інтеграції с CRM-системами.

14

ДЯКУЮ ЗА УВАГУ!