ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра інженерії програмного забезпечення автоматизованих систем

Пояснювальна записка

до магістерської роботи

на ступінь вищої освіти магістр

на тему: «РОЗРОБКА СИСТЕМИ ВІДСТЕЖЕННЯ КООРДИНАТ ОБ'ЄКТІВ, ОТРИМАНИХ ЗА ДОПОМОГОЮ МОДУЛІВ GSM, GPRS З ВИКОРИСТАННЯМ WEB-ТЕХНОЛОГІЙ»

Виконав: студент 6 курсу, групи ІСДМ-61

спеціальності

126 Інформаційні системи та технології

(шифр і назва спеціальності)

Олексієнко О.С.

(прізвище та ініціали)

Керівник Тучиш А.М.

(прізвище та ініціали)

Рецензент _____

(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем
Ступінь вищої освіти - «Магістр»
Спеціальність підготовки – 126 - Інформаційні системи і технології
Освітня програма «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІСТ
_____ К. П. Сторчак
"____" _____2022 року

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

Олексієнко Олександру Сергійовичу
(прізвище, ім'я, по батькові)

1.Тема роботи: : Розробка системи відстеження координат об'єктів, отриманих за допомогою модулів GSM, GPRS з використанням Web-технологій

Керівник роботи Тушич А. М. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від _____ № ___.
2. Строк подання студентом роботи _____
3. Вхідні дані до роботи:
Структура апаратної частини;
Python для IoT;
Науково-технічна література з питань, пов'язаних відстеження координат за допомогою модулів GSM, GPRS

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
4.1 Дослідження системи відстеження координат об'єктів.
4.2 Використання Python в системі відстеження координат об'єктів.
4.3 Реалізація MVP системи відстеження координат об'єктів.

5. Перелік графічного матеріалу
1. Мета і завдання.
2. Актуальність та практична цінність.
3. Дослідження системи відстеження координат об'єктів.

4. Апаратна частина системи. Опис SIM800L.
5. Структура веб-додатка.
6. Приклад реалізації системи відстеження координат.
7. Висновки.

6. Дата видачі завдання _____ року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури		
2	Дослідження системи відстеження координат.		
3	Вибір фреймворків Python для реалізації практичної частини		
4	Організація локального оточення для розробки практичної частини		
5	Реалізація та тестування практичної частини		
6	Розробка демонстраційних матеріалів (Презентація)		
7	Попередній захист роботи		
8	Подання роботи в деканат		

Студент _____ Олексієнко О.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Тушич А.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської роботи: 63 с., 39 рис., 1 дод., 18 джерел.

СИСТЕМА ВІДСТЕЖЕННЯ КООРДИНАТ ОБ’ЄКТІВ, АРХІТЕКТУРА, IoT, GSM, GSPRS, SIM800L, КОНТРОЛЬ, АВТОМАТИЗАЦІЯ, MQTT, PYTHON, FLASK, AT-commands, LINUX.

Об'єктом дослідження - веб-додаток для системи відстеження координат об’єкт.

Предметом дослідження є система відстежування та відображенням координат у веб-додатку за допомогою мови Python.

Мета роботи – є реалізація системи відстежування та відображенням координат у веб-додатку за допомогою мови Python.

Методи дослідження. Прикладне використання теоретичних навичок та знань для реалізації веб-додатка.

Для виконання даної задачі у магістерській роботі було реалізовано систему відстеження координат за допомогою технологій GSM, GPRS та мови програмування Python. Проаналізовано різні підходи та концепції. Проаналізовано та обрано основні елементи, які необхідно для реалізації проекту, а саме фреймворки, бібліотеки, операційні системи, апаратна частина.

ЗМІСТ

Стор.

ВСТУП

Актуальність теми. На сьогодні тема відстеження координат об’єктів не часто з'являється в обговорені на різних ресурсах. Дана тема має широкий спектр застосування на цей момент. Інженери створюють все нові та нові програмні та апаратні реалізації з кожним разом зменшуючи собі вартість того чи іншого продукту.

Розгортаючи систему відстеження координат об’єктів взаємодіють різні технології. Наприклад як GSM, GPRS, GPS, різні мови програмування та апаратна частина як трекери для відстежування до Ublox NEO-6M на базі Arduino.

Об'єктом дослідження є реалізація веб-додатка для системи відстежування координат об'єктів.

Предметом дослідження є система відстежування та відображенням координат у веб-додатку за допомогою мови Python.

Метою магістерської роботи є розробка веб-додатку для системи відстеження координат за допомогою технологій GSM, GPRS та мови програмування Python. Фреймворки Python використовуються для створення адмін панелі для авторизації користувача та відображення статистики та даних координат як на карті, так і в табличній формі.

1. ДОСЛІДЖЕННЯ СИСТЕМИ ВІДСТЕЖЕННЯ КООРДИНАТ ОБ'ЄКТІВ НА ОСНОВІ МОДУЛЯ SIM800L.

1.1 Структура системи відстеження координат об'єктів на основі технологій GSM, GPRS.

Телефонна мережа виникла задовго до Інтернету. Всі з'єднання в цій мережі здійснювалися аналоговими електричними схемами, і всі телефонні дзвінки були комутованими - тобто між абонентами необхідно встановлювати тимчасовий виділений ланцюг. Потім з'явилися модеми, які дозволили комп'ютерам обмінюватися даними по тому же телефонним аналоговим ланцюгам. Поступово телефонні комутатори були замінені маршрутизаторами, і нові телефонні мережі стали цифровими. Тепер абоненти з'єднуються віртуальними ланцюгами, і сценарії телефонних дзвінків не сильно відрізняються від сценаріїв електронної пошти чи чату: між абонентами встановлюється сеанс, відбувається обмін бітами та обмін повідомленнями. Відмінність між телефонним дзвінком та електронною поштою зараз міститься в протоколах, що застосовуються для кожного виду зв'язку, а не в структурі електричних кіл.

Однак телефонні компанії поверх послуг звичайної телефонії надають мережу ліній та маршрутизаторів, в якій першочергова увага приділяється голосовим послугам, що гарантує нам якісний сервіс, який ми не завжди отримуємо від сервісів, що працюють тільки з IP-телефонією.

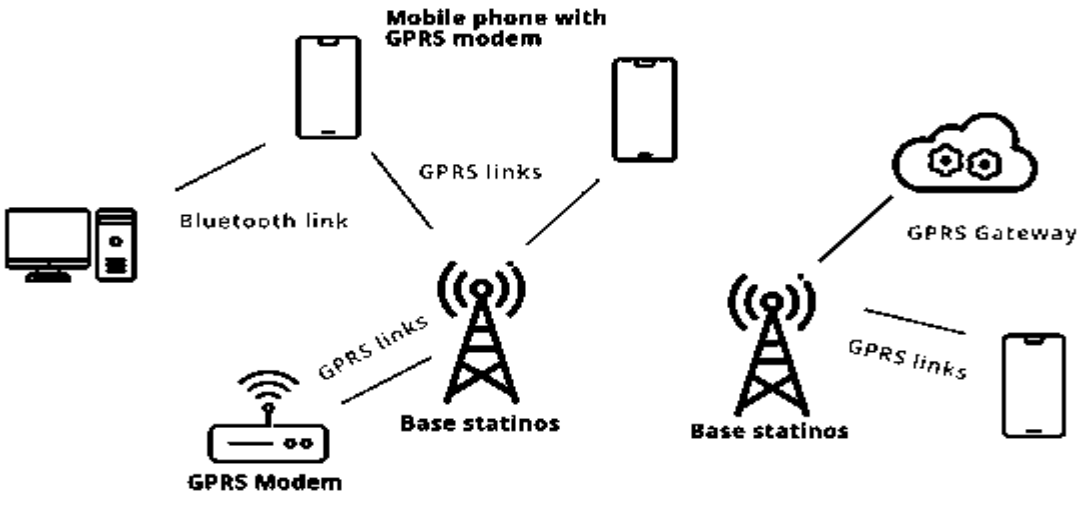


Рисунок 1.1 Структура «Розумного будинку»

GPRS можна використовувати для полегшення з'єднань, пов'язаних з Інтернет-протоколами, які забезпечують набір функцій, включаючи комерційні та корпоративні програми. Перед передачею інформація розбивається на окремі пакети та направляється через базову мережу та радіо. На стороні одержувача дані повторно підключаються.

GPRS може працювати як із симетричною, так і з асиметричною конфігурацією, тоді як частота для будь-якого напрямку визначається тим, який із 12 класів провайдерів із кількома слотами вибрано. Кількість часових інтервалів для кожного шляху визначається багатослотним класом обслуговування, для кожного часового інтервалу забезпечується теоретична швидкість з’єднання 21,4 Кбіт/с. Одним із найпростіших є клас обслуговування 1, який дозволяє одноразовий слот для кожного шляху. Клас обслуговування 12, безумовно, найефективніший, із чотирма часовими слотами в кожному напрямку. GSM-IP означає Інтернет-протокол мобільного зв’язку глобальної системи, а також іншу назву для GPRS. Це гарантує, що клієнти підключені, здійснюють аудіодзвінки та переглядають веб-сторінки. Цей метод забезпечує пакетний радіодоступ клієнтам множинного доступу з рівним часовим поділом (TDMA). GPRS також дозволяє мережевим операторам розгортати основну архітектуру на основі IP для інтегрованих додатків аудіо та даних, які можна використовувати та покращувати для мереж 3G.

Ідея якості обслуговування (QoS) включена в GPRS. Це стосується здатності адаптувати послугу до вимог програми. Застосовуються наступні умови QoS – пріоритет, надійність, затримка та пропускна здатність. Для інтеграції GPRS в архітектуру GSM необхідне включення нових вузлів мережі, які називаються GSN (вузли підтримки GPRS), розташовані в магістральній мережі:

- SGSN (Serving GPRS Support Node) — це маршрутизатор, який контролює розташування найближчих станцій і пропонує інтерфейс передачі пакетів до порталу GGSN.

- Шлюз, який з’єднується з іншими мережами передачі даних, — це GGSN (вузол підтримки шлюзу GPRS). GGSN відповідає за доставку IP-адреси до мобільних терміналів протягом усього з’єднання.

У порівнянні з послугами з комутацією каналів передача пакетів GPRS пропонує кращі рахунки споживачам. Коли йдеться про послуги з комутацією каналів, вартість визначається довжиною з’єднання. Тепер припустімо, що ви використовуєте систему відстеження активів або дистанційний датчик. Бажано, щоб ці гаджети були «завжди увімкненими». З іншого боку, ці продукти призначені лише для передачі або отримання інформації дещо нерегулярно. Кожен абонент несе відповідальність за всю вартість ефірного часу, в тому числі в періоди, коли пакети не відправлялися.

GPRS технічно підходить для більшості випадків використання IoT, але не для всіх. 2G або 3G з GPRS зазвичай є теоретично прийнятним, якщо все, що вам потрібно, це повільна періодична передача даних. Однак для додатків, що інтенсивно передають дані, GPRS впливає на наявну ємність клітинки системи. Є лише кілька радіоресурсів, які можна використовувати для різних цілей. Голосові дзвінки та дзвінки GPRS, наприклад, використовують подібні послуги мережі. Величина впливу визначається кількістю часових інтервалів, встановлених лише для використання GPRS.

Ризики безпеки та помилки продовжують турбувати виробників та користувачів технології. Одна з головних проблем "розумного будинку" це те що хакери можуть отримати доступ до системи та керувати побутовою технікою та іншими приладами й датчиками за допомогою Інтернету.

GPRS означає загальну пакетну радіопередачу, і це була перша мобільна технологія, яка спробувала підключити наші телефони до Інтернету. Швидкість зазвичай становила від 40 до 128 Кбіт/с у пізніших версіях, а сама технологія була досить дивною, майже такою ж, як підключення та використання модему. Ви платили за GPRS не за хвилини, а за МБ, що було свіжим, але знову ж таки, наші телефони просто не могли реально використовувати дані, як ми сьогодні. WAP був

ранньою формою мобільного Інтернету, частково заснованого на HTML, але навіть він був поганим, і по суті все, що я пам’ятаю, використовував GPRS, це підключити мій iPAQ КПК до Інтернету, щоб читати мої електронні листи! 2.5G або EDGE, як його було відомо, був наступним. Edge, по суті, була тією ж технологією, що й GPRS, але вона була як би дуплексною через кілька каналів, тож це означало, що теоретично ви могли бачити швидкість до 400 Кбіт/с. Однак справжні зміни в 2.5G відбулися в тому, як він працює. 2G не дозволить вам телефонувати та використовувати дані одночасно, однак 2,5G дозволить, тож уперше ви можете розмовляти, але також завантажувати дані у фоновому режимі. Це заклало б основу для того, що буде далі, що змінить мобільний світ, яким ми його знали.

У топології 3G використовувалася комутація каналів для трафіку мови та CMC та пакетна комутація для даних. Комутація каналів докорінно відрізняється від пакетної комутації. Комутація каналів починається в оригінальній телефонній комутованій мережі. Вона використовує виділений канал зв'язку, який є з'єднання між вузлом-адресантом і вузлом-адресатом, це з'єднання протягом усього сеансу обміну інформацією може використовуватися тільки двома зазначеними вузлами. У мережі з пакетною комутацією повідомлення розбиваються на невеликі частини (які називаються пакетами у разі даних IP) і знаходять найкоротший шлях від адресанта даних до адресата. У заголовку, яким оснащений пакет, обов'язково вказана адреса вузла-отримувача. Архітектура 3GPP LTE називається "Еволюція системної архітектури" (SEA), і її спільною метою є надати спрощену архітектуру, повністю побудовану на IP. Вона здатна підтримувати високошвидкісну передачу даних із низькою затримкою в мережах з радіодоступом (RANs). У 8 релізі 3GPP roadmap було введено стандарт LTE. Такі мережі складаються з компонентів із пакетною комутацією на базі протоколу IP, що означає, що голосові дані надсилаються у вигляді цифрових IP-пакетів. Це є ще однією корінною відмінністю від традиційної мережі 3G.

Типова 4G-LTE-мережа складається з трьох компонентів: клієнт, радіомережа та базова мережа. Клієнтом є радіопристрій користувача. Радіомережа представляє прямий зв'язок між базовою мережею і клієнтом і включає

радіообладнання, таке як вежа. Базова мережа представляє інтерфейс керування постачальником і може підтримувати одну або більше радіомереж.

Важливо розуміти, як системи стільникового зв'язку працюють з численними користувачами даних та голосового зв'язку. Існує кілька стандартів, схожих з концепціями, описаними для WPAN та WLAN, з якими слід ознайомитися. До того, як 3GPP мала можливість підтримувати LTE, існувало безліч стандартів для технології стільникового зв'язку, зокрема, для пристроїв GSM і CDMA. Слід зазначити, що ці технології не сумісні один з одним, починаючи з інфраструктури та закінчуючи самим пристроєм:

- множинний доступ із поділом частот (FDMA) – поширений в аналогових системах, але нині рідко використовують у цифрових доменах. Це техніка, згідно з якою спектр ділиться на частоти та розподіляється серед користувачів. Один трансівер у будь-який проміжок часу приписується до каналу зв'язку. Канал, відповідно,закрито для інших дзвінків, доки не завершиться початковий дзвінок або він не буде підключений до іншого каналу. Повністю дуплексний режим передачі вимагає наявності двох каналів зв'язку: одного передачі, іншого прийому;
- множинний доступ із кодовим поділом (CDMA) – заснований на технології розширення спектра. CDMA збільшує пропускну спроможність спектру, дозволяючи користувачам зайняти всі канали зв'язку одночасно. Передача відбувається по всьому радіодіапазону, і кожному сигналу надається унікальний код для його відокремлення із загальної какофонії сигналів по всьому спектру. CDMA допускає «м'який хендовер», що означає, що термінали можуть підключатися відразу до кількох базових станцій одночасно. Домінуючий радіоінтерфейс для мобільного зв'язку третього покоління був розроблений американською компанією з розробки та дослідження бездротових засобів зв'язку Qualcomm як CDMA2000 та призначався для 3G. Через свою пропрієтарну природу він не був запроваджений

повсюдно і використовується менш ніж 18% світового ринку. Був введений у США, де компанії Verizon та Sprint були його постійними постачальниками.

- множинний доступ із тимчасовим поділом каналів (TDMA) – у цій моделі пропускна спроможність спектру збільшується шляхом поділу кожної частоти на часові інтервали. TDMA дозволяє кожному користувачеві отримати доступ до всього радіочастотного каналу на короткий період дзвінка. Інші користувачі мають доступ до цієї частоти в різні проміжки часу. Базова станція постійно перемикається з користувача користувача, що перебуває на даному каналі зв'язку.
- TDMA є домінуючою технологією мобільних мереж другого покоління. Організація GSM запровадила TDMA як модель множинного доступу. Вона властива для 4 різних діапазонів частот: 900 МГц/1800 МГц у Європі та Азії та 850 МГц/1900 МГц у Північній та Південній Америці;

Деякі пристрої та модеми досі підтримують GSM/LTE або CDMA/LTE. GSM та CDMA несумісні один з одним. GSM/LTE та CDMA/LTE можуть бути сумісні, однак якщо вони підтримуватимуть діапазон частот LTE. У більш старих моделях інформація у звуковій формі передається за допомогою спектру 2G або 3G, які значною мірою відрізняються від CDMA та GSM (TDMA). Дані також несумісні, оскільки дані LTE передаються діапазонах частот 4G.

Існує 55 діапазонів частот LTE, на що частково вплинула фрагментація спектра та ринкові стратегії. Розподіл смуг частот LTE також є нагородою аукціону частот, який проводиться державою. LTE також ділиться на 2 категорії, які несумісні одна з одною:

- дуплекс з тимчасовим поділом (TDD) – TDD використовує загальний діапазон частот передачі та прийому даних. Напрямок передачі визначається тимчасовими слотами;

- дуплекс із частотним поділом (FDD) – у конфігурації FDD базова станція (eNodeB) та абонентський пристрій (UE) відкривають два діапазони частот для прийому та передачі даних. Прикладом може бути смуга частот LTE-13, у якої діапазон передачі даних варіюється від 777 до 787 МГц, а діапазон прийому даних від 746 до 756 МГц. Передача та прийом даних можуть відбуватися одночасно.

Існує режим комбінування модулів TDD і FDD, який об'єднує їх в одному модемі і допускає використання кількох несу.

У 4G LTE абоненту надається постачальник або оператор, відомий як Мережа зв'язку загального користування наземних мобільних об'єктів (PLMN). Якщо абонент знаходиться в мережі PLMN свого постачальника, він або вона знаходиться в домашній PLMN-мережі. Якщо ж абонент переміщається до іншої мережу, яка знаходиться за межами його домашньої мережі (наприклад, під час поїздок за кордон), вона називається гостьовою PLMN. Абонент підключає своє обладнання до гостьової мережі, що потребує ресурсів E-UTRAN, MME, SGW та PGW нової мережі. PGW надає доступ до Інтернету. У цей момент послуги роумінгу починають впливати на тарифний план. Оплата за роумінг стягується гостьовою мережею та вказується у рахунку клієнта. Нижче ви побачите ілюстрацію до цієї архітектури. Праворуч зображено модель переміщення абонента в гостьову мережу та розподіл функціональних можливостей між гостьовою мережею E-UTRAN та вдосконаленим пакетним ядром (EPC), а також показано шлях до домашньої мережі. Інтерфейс S5 використовується, якщо клієнт та постачальник знаходяться в одній мережі, а S8 – коли клієнт перетинає межі інших мереж.

5G (або 5G-NR для нового радіо) є стандартом для IP-комунікацій нового покоління, розробленим та призначеним для заміни 4G LTE. Тим не менш, він використовує деякі технології 4G LTE, але має суттєві відмінності та нові можливості. По 5G написано так багато матеріалу, що він затьмарює навіть поточний матеріал Cat-1 і Cat-M1. Зокрема, 5G обіцяє надати суттєві можливості

для IoT, комерційних, мобільних та транспортних застосувань. 5G також покращує пропускну здатність, латентність, щільність та вартість для користувача. Замість того, щоб створювати різні стільникові послуги та категорії для кожного варіанту використання, 5G намагається стати одним зонтичним стандартом, щоб обслуговувати їх усіх. Тим часом, 4G LTE залишатиметься домінуючою технологією для покриття стільникового зв'язку і продовжуватиме розвиватися. 5G не є продовженням еволюції 4G він походить від 4G, але є новий набір технологій. У цьому розділі ми розглянемо лише ті елементи, які відносяться до випадків використання IoT або заслуговують на увагу, і можуть стати частиною специфікації 5G.

Визначення відстані надає інформацію про те, як далеко знаходиться об'єкт від точки, з якої виконується визначення, в одному вимірі, але не надає повної інформації про розташування об'єкта. Відстань між вихідною точкою та цільовим об'єктом визначається колом із центром у вихідній точці (або сферою — у разі трьох вимірів). Наш об'єкт може знаходитись у будь-якій точці цього коло. У глобальній системі позиціонування GPS використовується трилатерація на основі мережі супутників Землі. Розташування кожного з супутників можна визначити за траєкторією його польоту та поточним часом. Кожен супутник передає сигнал часу свого годинника, який приймається приймачами GPS. Коли приймач отримує сигнал як мінімум від трьох супутників, він може обчислити своє місцезнаходження на основі різниці в часі між передачею і прийомом. Більшість приймачів GPS використовують сигнал мінімум від шести супутників, щоб мінімізувати помилки місцезнаходження. Системи визначення місцезнаходження мобільних телефонів, такі як Wireless E911, обчислюють приблизне розташування телефонів подібним чином, вимірюючи відстань від антен декількох базових станцій на основі різниці часу між моментами приходу сигналів від цих антен.

Послідовний протокол NMEA 0183 працює зі швидкістю 4800 бітів за секунду, слово складається з 8 бітів, перевірка на парність не проводиться, є один столовий біт (формат 4800-8-N-1). Більшість приймачів видають дані,

використовуючи рівні RS-232 або TTL. Залучений у нашому прикладі приймач US GlobalSat EM-406a видає дані у форматі NMEA, використовуючи рівні TTL 5.

Багато сучасних мобільних телефонів оснащені приймачем GPS і, начебто, приймають сигнал майже завжди, що може змусити вас задуматися - чому ваш приймач не може налаштуватись на сигнал також швидко. Слід пам'ятати, що для визначення розташування мобільні телефони зазвичай використовують як GPS, так і базові станції мережі. Тому, якщо вони не можуть отримати сигнал від достатньої кількості супутників, вони зазвичай визначають своє місце розташування щодо найближчих базових станцій і використовують цю інформацію, щоб надати вам приблизні координати. Протокол NMEA 0183 є лише одним з багатьох протоколів, які використовуються для отримання даних GPS. Незалежно від того, який інший протокол вони можуть використовувати, всі вони зазвичай працюють за протоколом NMEA. Тому буде корисно знати цей протокол, незалежно від того, якими інструментами GPS ви користуєтесь.

1.2 Протокол TCP/IP в M2M.

M2M може зв'язуватися через WAN з виділеним сервером або системою без інкапсульованого протоколу. За визначенням IoT заснований на зв'язку між кінцевими точками та службами, причому інтернет є загальною мережевою основою.

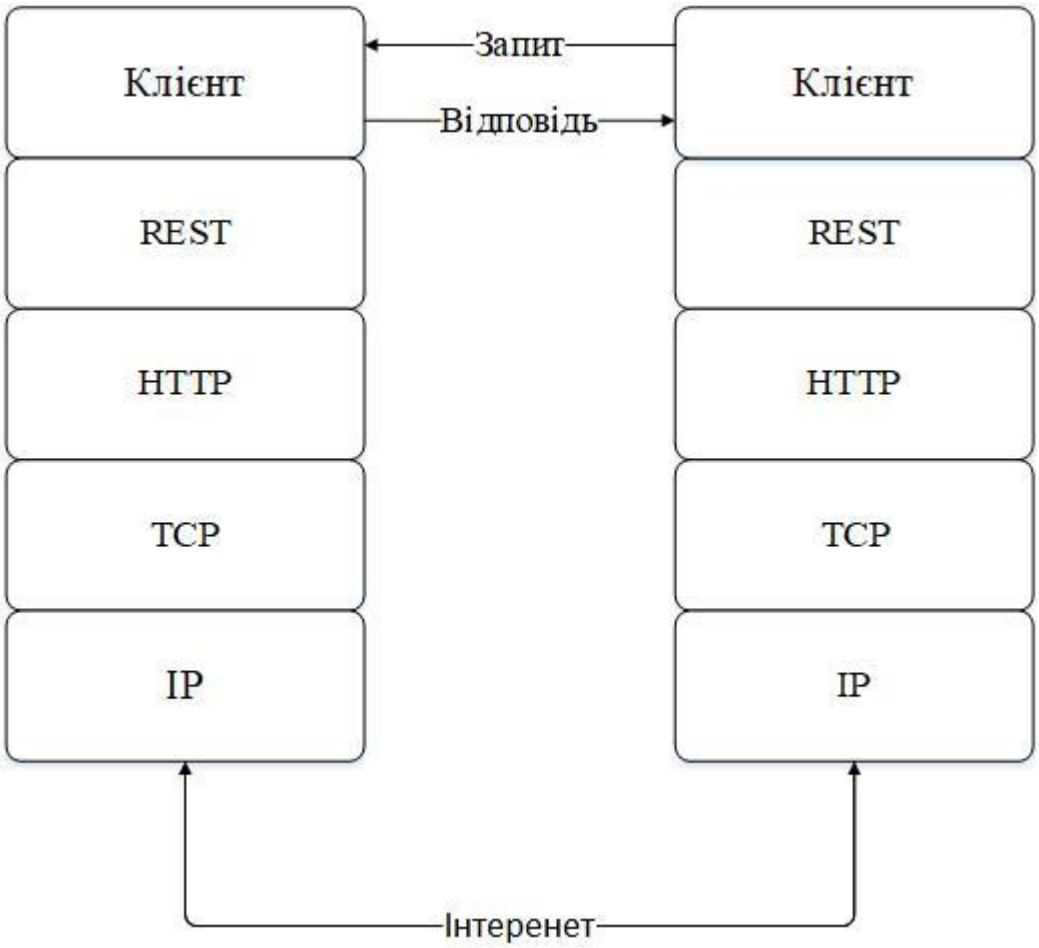


Рисунок 1.2 Схема RESTful

A9G можна використовувати в широкому діапазоні IoT і ідеально підходить для програм Інтернету речей для дому автоматизація, промислове бездротове керування, носіння електроніка, бездротові пристрої визначення місцезнаходження, сигнали бездротової системи визначення місцезнаходження та інші IoT програми.

Пакет A9G SMD через стандартний SMT обладнання для досягнення швидкого виробництва продукції, спеціально для автоматизації, масштабний, недорогий сучасні методи виробництва для зручності а різновид апаратного терміналу Інтернету речей програми.

Служби RESTful повинні мати єдиний інтерфейс для доступу до ресурсів, і, як випливає з назви, інтерфейс API для системи повинен бути єдиним для всієї системи. Логічна система URI з уніфікованими способами отримання та маніпулювання даними – це те, що полегшує роботу з REST. HTTP/1.1 надає набір

методів для роботи з ресурсами на основі іменників; Для цієї мети методи зазвичай називаються дієсловами. В архітектурі REST існує концепція безпечних і ідемпотентних методів. Безпечні методи – це ті, які не змінюють ресурси, як метод GET або HEAD. Ідемпотентний метод — це метод, який дає однаковий результат незалежно від того, скільки разів він виконується.

Сервіси RESTful зосереджені на ресурсах і наданні доступу до ресурсів. Ресурс можна легко розглядати як об’єкт в ООП. Перше, що потрібно зробити під час проектування служб RESTful, це визначити різні ресурси та визначити зв’язок між ними. Представлення — це машинозчитуване пояснення, що визначає поточний стан ресурсу. Після визначення ресурсів представлення — це наступний курс дій. REST надає нам можливість використовувати будь-який формат для представлення ресурсів у системі. На відміну від SOAP, який обмежує нас використовувати XML для представлення даних, ми можемо використовувати JSON або XML. Зазвичай JSON є кращим методом представлення ресурсів, які викликаються мобільними або веб-клієнтами, але XML можна використовувати для представлення більш складних ресурсів.

Кешування — це техніка, яка зберігає копію даного ресурсу та повертає її за запитом, заощаджуючи додаткові виклики БД і час обробки. Це можна зробити на різних рівнях, таких як клієнт, сервер або проксі-сервер проміжного програмного забезпечення. Кешування є важливим інструментом для підвищення продуктивності API та масштабування програми; однак, якщо не керувати належним чином, це призводить до того, що клієнт отримує старі результати. Кешування в REST API контролюється за допомогою заголовків HTTP. Заголовки кешу були важливою частиною специфікацій HTTP-заголовків і були важливою частиною ефективного масштабування веб-служб. У специфікації REST, коли безпечний метод використовується для URL-адреси ресурсу, зазвичай зворотний проксі кешує результати для використання кешовані дані, коли той самий ресурс запитується наступного разу.

1.2.1 Можливості вбудованого стека протоколів TCP/IP в GSM/GPRS модулях SIM800.

GSM-модуля SIM800L на прикладах з платою Arduino Uno, а саме - відмінності модулів один від одного, підключення модуля, взаємодія з ним за допомогою AT-команд, підключення мікрофона та колонок, здійснення дзвінків, прийом та відправка SMS та USSD-запитів, сплячий режим модуля — занурення та пробудження, розпізнавання DTMF, визначення GPS-координат, FM-радіо, прийом та надсилання даних по GPRS.

Існує два режими з’єднання для додатків TCP/IP серії SIM800: одинарне з’єднання та кілька з’єднань. У режимі одного підключення SIM800 може працювати як у прозорому, так і в непрозорому режимі. І в цих двох режимах передачі SIM800 серії можна налаштувати як клієнт TCP/UDP або сервер TCP. У режимі кількох підключень серія SIM800 може працювати лише в непрозорому режимі. У цьому режимі SIM800 серії може працювати як абсолютний клієнт TCP/UDP, який може встановити загалом 6 з'єднань. У цьому режимі він також може бути налаштований як один сервер TCP, що дозволяє під'єднати 5 клієнтів TCP/UDP. І TCP-сервер також може діяти як клієнт, встановлюючи 5 підключень до одного віддаленого сервера. Структура програми TCP/IP наведена нижче.

Серія SIM800 підтримує прозорий режим, який забезпечує спеціальний режим даних для отримання та надсилання даних прикладним завданням TCP/IP. Після встановлення з’єднання в прозорому режимі модуль перейде в режим даних. Усі дані, отримані з послідовного порту, розглядатимуться як пакет даних, який буде передано пізніше, так само всі дані, отримані з віддаленої сторони, будуть надіслані безпосередньо на послідовний порт. У прозорому режимі всі AT-команди недоступні. Надаються методи для перемикання між режимом даних і командним режимом. Після перемикання в командний режим усі AT-команди можна використовувати знову.

Щоб увімкнути прозорий режим, для команди AT+CIPMODE має бути встановлено значення 1. У прозорому режимі для налаштування режиму передачі використовується команда AT+CIPCCFG, яка має 7 параметрів NmRetry, WaitTm, SendSz, Esc, Rxmode, RxSize, Rxtimer.

- NmRetry: кількість повторних спроб для IP-пакета.

- WaitTm: кількість інтервалів у 200 мс для очікування послідовного введення перед надсиланням пакета.
- SendSz: Розмір у байтах блоку даних, який буде отримано з послідовного порту перед надсиланням.
- Esc: чи вмикається escape-послідовність, за замовчуванням TRUE.
- Rxmode: чи встановлювати часовий інтервал під час виведення даних із послідовного порту.
- RxSize: довжина вихідних даних кожного разу, значення за замовчуванням 1460.
- Rxtimer: інтервал часу (мс) для очікування, поки послідовний порт знову виведе дані. Значення за замовчуванням: 50 мс.

Щоб перемикнутися з режиму даних у режим команд, доступні такі методи: послідовність можна використовувати, якщо четвертий параметр AT+CIPCCFG має значення TRUE. Екранна послідовність за замовчуванням — +++, і для використання цієї послідовності має бути 1000 мс період простою перед цією послідовністю та 1000 мс період простою після цієї послідовності. Крім того, інтервал між кожним + не повинен перевищувати 1000 мс, інакше він буде розглядатися як дані TCP/IP. Лінія DTR послідовного порту також може бути використана. Щоб використовувати цей метод, спочатку слід встановити AT&D1. Потягніть лінію DTR на землю щонайменше на 1 секунду, а потім потягніть вгору, модуль буде перемикнути з режиму даних у режима команди, і буде повернено OK, що вказує на те, що модуль увімкнено командний режим.

Для TCP-клієнтського з’єднання, якщо віддалений сервер закриває з’єднання, модуль автоматично повертається в командний режим.

Для з’єднання з TCP-сервером, якщо віддалений клієнт закриває з’єднання, модуль автоматично повертається в командний режим.

Якщо модуль дезактивовано з контексту PDP (+PDP DEACT) під час передачі даних, модуль автоматично повернеться в командний режим. Команда ATO може

бути використана для перемикання модуля з командного режиму в режим даних, якщо з'єднання активне, і знову буде повернено CONNECT.

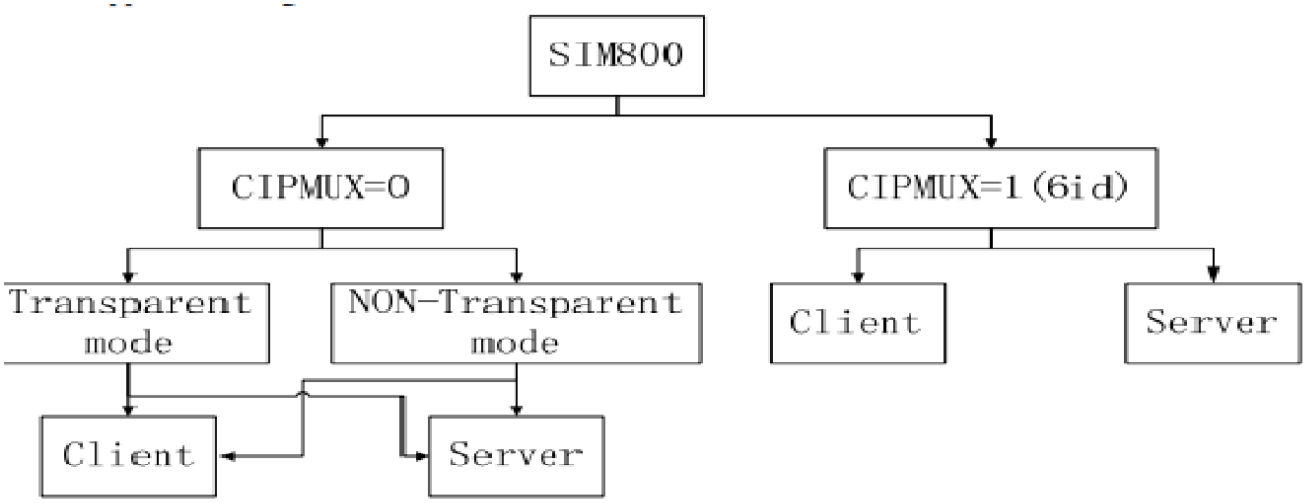


Рисунок 1.3 SIM800 TCP/IP

У режимі кількох підключень серія SIM800 може працювати як клієнт TCP/UDP або сервер TCP. Як клієнт, він може встановити загалом 6 підключень до віддаленого сервера, як TCP, так і UDP. Як TCP-сервер, він дозволяє віддаленим клієнтам підключатися. Підключення TCP/UDP до верхніх віддалених серверів. Всього підтримується 5 доступних підключень, при цьому сам сервер займає одне підключення.

1.2.2 Застосування SIM800L.

Розпіновка GSM/GPRS модуля SIM800L представлена на наступному малюнку. Як видно, він має 12 контактів: NET, VCC, RST, RXD, TXD, GND, SPK-, SPK+, MIC-, MIC+, DTR, RING.

- NET – до цього контакту можна припаяти спіральну антену, яка постачається разом із модулем.
- VCC – через цей контакт на модуль подається напруга живлення, яка може становити від 3.4 до 4.4 Вольт.

- RST – контакт апаратного скидання модуля SIM800L. Для скидання модуля необхідно подати цей контакт рівень low тривалістю щонайменше 100ms.
- RXD – контакт приймання (RX) модуля для послідовного зв'язку.
- TXD – контакт передачі (TX) модуля для послідовного зв'язку.
- GND – контакт підключення модуля до загального дроту схеми (землі).
- RING – контакт вказівника дзвінків модуля. У випадку на цьому контакті підтримується рівень high. Також його можна запрограмувати формування імпульсу прийому SMS.
- DTR – цей контакт використовується для переведення модуля SIM800L у режим сну. Подача на цей контакт рівня High переводить його в сплячий режим і відключає в ньому послідовний зв'язок.
- MIC+- – ці два контакти можна використовувати для підключення зовнішнього мікрофона до модуля.
- SPK+- – ці два контакти можна використовувати для підключення зовнішнього гучномовця (динаміка) до модуля.

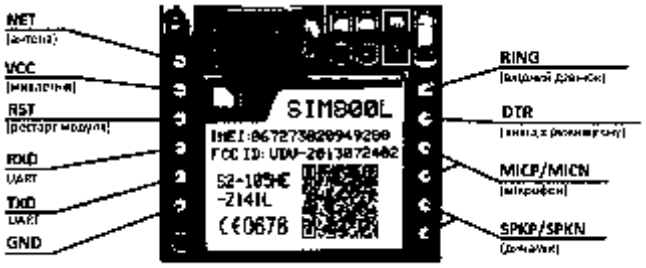


Рисунок 1.4 Модуль SIM800L

На передній стороні модуля розташований конектор UFL і, власне, сам модуль SIM800L. Також на ній розташовані конденсатори, обмежувач резистор 1 кОм для світлодіода. Також на цій стороні модуля розташований великий танталовий конденсатор 100uF, 16V. На зворотному боці модуля розташований тримач SIM карт.

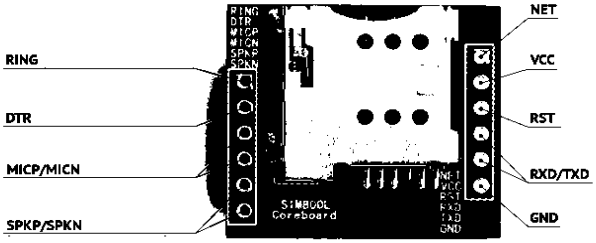


Рисунок 1.5 Модуль SIM800L кріплення SIM карти

На передній стороні модуля SIM800L розташований світлодіод-індикатор, він блимає по-різному залежно від режиму роботи модуля.

Блимає кожну секунду це означатиме, що модуль запущений в роботу, але поки не приєднатися до мережі стільникового зв'язку.

Блимає кожні 2 секунди це означає, що GPRS з'єднання готове до роботи.

Блимає кожні 3 секунди це означає, що модуль підключений до мережі і може передавати/приймати голос та SMS.

SIM800L підтримує лише 2G сервіси. SIM карти 4G у модулі працюють, але лише у режимі 2G.

Модулі SIM800L упаковані під вакуумом і гарантовано зберігаються протягом 6 місяців без відкриття або витоку під такі умови: температура навколишнього середовища нижче 40 °C, а вологість повітря менше ніж 90%. Якщо умова відповідає одній із наведених нижче умов, модулі повинні бути заздалегідь достатньо пропечені повторне паяння, а умови випікання показано в таблиці нижче інакше модуль буде під загрозою незворотне пошкодження під час паяння повторним потоком.

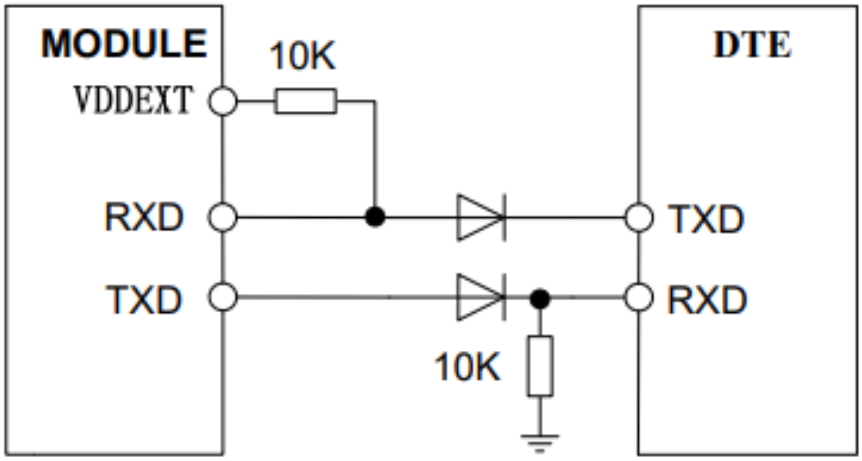


Рисунок 1.6 Діодна схема ізоляції

Сигнал GSM може створювати перешкоди для аудіо через зв’язок або провідність. Зв'язковий шум можна відфільтрувати, додавши конденсатор 33 пФ і 10 пФ над аудіо лініями. Конденсатор місткістю 33 пФ може усунути шум від GSM850/EGSM900 МГц, тоді як конденсатор місткістю 10 пФ може усунути шум від частоти DCS1800/PCS1900 МГц. Шум зчеплення повинен мати щось пов'язане з компонуванням друкованої плати. За деяких сценаріїв шум TDD від частоти GSM850/EGSM900 МГц сильно впливає, але інша історія стосується частоти DCS1800/PCS1900 МГц, тому замовник повинен розробити це фільтраційне рішення відповідно до результатів польових випробувань. Антена GSM є ключовим зв'язком, що заважає джерелу шуму TDD. При цьому зверніть увагу на розташування аудіоліній, які повинні бути подалі від радіочастотного кабелю та антени та контакту VBAT. Байпасний конденсатор для фільтрації слід розмістити біля модуля, а іншу групу потрібно розмістити біля роз’єму. Провідний шум в основному спричинений падінням VBAT. Якщо аудіо PA живиться безпосередньо від VBAT, тоді з виходу динаміків легко буде чути дешевий шум. Тому краще поставити великий конденсатор і феритову кульку біля аудіовходу PA. Безумовно, шум TDD має щось спільне з сигналом GND. Якщо сигнал GND поганий, багато високочастотних шумів створюватимуть перешкоди для мікрофона та динаміка

через байпасний конденсатор. Тому під час компонування друкованої плати потрібно подбати про хороше заземлення.

1.2.3 Опис AT команд SIM800L.

Команди AT дозволяють контролювати все навколо модему, пристрою та SIM-карти. AT-команди зазвичай мають стандартизоване синтаксичне правило - більшість команд мають три різні типи: запис, читання та перевірка. Крім того, є й інші, які можна виконати лише для отримання інформації. Усі стандартні AT-команди починаються з AT+<command>.

Оскільки AT-команди стандартизовані в 3GPP TS 27.005 і 27.007, і всі виробники повинні їх реалізувати, можуть бути відмінності залежно від типу модема та постачальника. Залежно від типу модема – наприклад, деякі команди енергоощадження доступні лише в новіших версіях, які також мають відповідні функції. Параметри також можуть відрізнятися, оскільки модем не підтримує цю конкретну конфігурацію типу радіозв'язку або код помилки.

Часто виробники вводять додаткові пропрієтарні AT-команди, які або мають нову функціональність, або вдосконалюють існуючі AT-команди. Хоча ці команди можуть бути потужними, необхідно враховувати мікропрограмне забезпечення від різних виробників.

Командний рядок, який використовується в кожному модемі, починається з «AT», інакше «at», тому ці команди називаються AT-командами. Є багато команд, які використовуються для керування модемами (дротовий комутований зв'язок), як-от ATD – набір номера, ATA – відповідь, ATH – керування підключенням і ATO – повернення до онлайнового стана даних. Вони підтримуються такими модемами, як мобільні телефони, GSM або GPRS. Є кілька команд, які підтримують GSM. Ці команди, які використовуються для GSM, в основному включають команди на основі SMS, такі як AT+CMGS, AT+CMSS, AT+CMGL & AT+CMGR. Тут префікс AT у цих командах інформує модем про початок командного рядка

Для контроля виклику:

- Команда ATA є командою відповіді.
- ATH використовується для завершення виклику.
- ATD — це команда набору.
- Команда ATM використовується для перевірки режиму динаміка.
- Команда ATT використовується для налаштування тонального набору за замовчуванням.
- Команда ATL використовується для перевірки гучності динаміків.
- Команда AT+CSTA використовується для вибору типу адреси.
- Команда ATP використовується для фіксації імпульсного набору за замовчуванням ATO — це команда Go on-line.
- Команда AT+CRC використовується для кодів результатів стільникового зв’язку.

Для контроля картки даних:

- Для ідентифікації використовується команда ATI .
- Команда AT&F використовується для відновлення заводських налаштувань .
- Команда ATZ використовується для нагадування збереженого профілю.
- Команда AT&W використовується для збереження параметрів у вказаному профілі Команда AT+CSTA використовується для вибору типу адреси.
- Команда AT+GMM використовується для визначення моделі запиту.
- Команда AT&V використовується для перевірки активної конфігурації.
- Команда AT+CLCK використовується для блокування об'єкта.
- Команда AT+GCAP використовується для завершення запиту списку можливостей Команда AT&Y використовується для вибору параметра включення.
- Команда AT+GMR використовується для визначення перегляду запиту.
- AT+COLP — це презентація розпізнавання підключеної лінії.

- AT+GMI — це команда запиту ідентифікації виробника.
- Команда AT+GSN використовується для запиту номера IMEI продукту.

Команди, які використовуються для керування телефоном, в основному включають наступне.

- Команда AT+CBC використовується для зарядки акумулятора.
- Команда AT+CGMR використовується для визначення перегляду запиту.
- Команда AT+CGSN використовується для запиту серійного номера продукту Команда AT+CMEE використовується для повідомлення про помилку мобільного обладнання.
- Команда AT+CPBF використовується для пошуку записів телефонної книги.
- Команда AT+CPBR використовується для читання записів телефонної книги Команда AT+CPBS використовується для вибору місця зберігання пам'яті телефонної книги .
- Команда AT+CSCS використовується для вибору набору символів TE .
- Команда AT+CPBW використовується для запису телефонної книги.
- Команда AT+CGMM використовується для визначення моделі запиту.
- Команда AT+CGMI використовується для визначення виробника запиту. AT+CSQ — це команда якості сигналу .
- Команда AT+CPAS використовується для перевірки стану активності телефону.

Команди, які використовуються для інтерфейсу комп’ютерних даних, в основному включають наступне.

- Команда ATQ призводить до придушення коду .
- Команда ATX використовується для вибору діапазону відповіді .
- Команда AT+ICF використовується для кадрування символу DTE-DCE .
- Команда AT&Q використовується для визначення опції режиму зв’язку .
- Команда AT&C використовується для визначення використання DCD .

- Команда AT&D використовується для визначення використання DTR .
- Команда ATV є визначенням формату відповіді
- Команда AT+IFC використовується для керування локальним потоком для DTE-DCE Команда AT&K використовується для вибору керування потоком
- Команда AT&S використовується для визначення опції для DSR
- Команда AT+IPR використовується для встановлення швидкості DTE

Команди, які використовуються для параметра мережевого зв'язку, в основному включають наступне.

- Команда ATB є опцією для стандарту зв’язку
- Команда AT+DS використовується для стиснення даних
- Команда AT+CEER використовується для розширення звіту про помилку
- Команда AT+CBST використовується для вибору типу носія
- Команда AT+CRLP є протоколом радіоканалу

Команди, які використовуються для текстового режима SMS, в основному включають наступне.

- Команда AT+CSMS використовується для вибору служби повідомлень.
- Команда AT+CSMP використовується для фіксації параметрів текстового режиму.
- Команда AT+CMGF використовується для форматування повідомлення.
- Команда AT+CPMS використовується для вибору сховища повідомлень.
- Команда AT+CRES використовується для відновлення налаштувань.
- Команда AT+CNMI використовується для вказівки нового повідомлення для ТЕ.
- Для адреси центру обслуговування використовується команда AT+CSCA.
- Команда AT+CSCB використовується для вибору типів повідомлень стільникового мовлення.

- Для читання повідомлення використовується команда AT+CMGR.
- Команда AT+CSDH використовується для ілюстрації параметрів текстового режиму .Команда AT+CSAS використовується для збереження налаштувань.
- Команда AT+CMSS використовується для надсилання повідомлення зі сховища.

Для видалення повідомлення використовується команда AT+CMGD .

- Команда AT+CMGL використовується для переліку повідомлень .
- Команда AT+CMGS використовується для надсилання повідомлення .
- Команда AT+CMGW використовується для запису повідомлення в пам’ять.

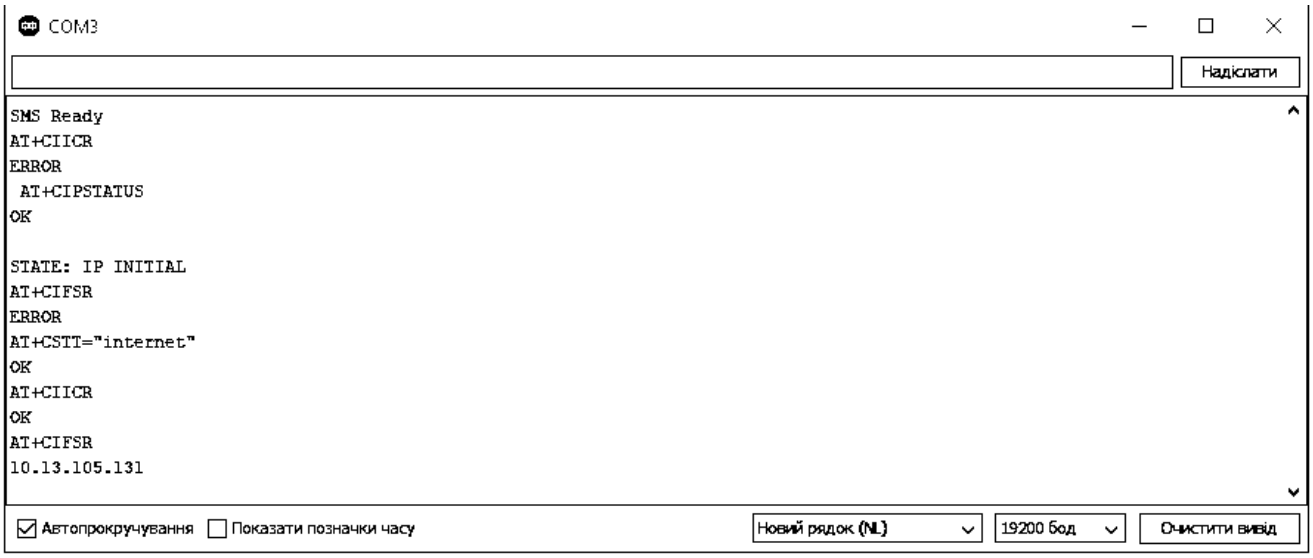


Рисунок 1.7 Вікно монітора порта

1.3 Програми u-center та m-center.

u-center — це інтуїтивно зрозуміле програмне забезпечення для оцінки GNSS простий у використанні, персоналізований і сумісний з провідними u-blox технології. u-center 2, наступне покоління програмного забезпечення, підтримує платформу u-blox M10. Оптимізований користувацький досвід u-center 2 включає персоналізовані робочі простори та адаптивні елементи вікон. Програвач журналів забезпечує легку навігацію на основі повідомлень і часу з регульованою швидкістю

відтворення та імпортом файлу журналу u-center. Швидке налаштування продукту u-center 2 дозволяє користувачам визначати або застосовувати конфігурації продуктів GNSS для конкретних випадків використання. Зберігати, відновлювати або ділитися конфігураціями між різними продуктами та користувачами легко. Програмне забезпечення підтримує оцінку продукту з вибором переглядів для спостереження за статичною та динамічною поведінкою підключеного приймача u-blox GNSS. Програмне забезпечення дозволяє легко налаштувати та оцінити сервіси GNSS u-blox, такі як AssistNow. Надаються регулярні оновлення u-center 2, щоб гарантувати, що програмне забезпечення завжди має найновіші функції та попередньо визначені конфігурації, а також для додавання підтримки останньої версії мікропрограми приймачів GNSS u-blox. Користувач отримує інформацію під час запуску та може встановити оновлення одним клацанням миші.

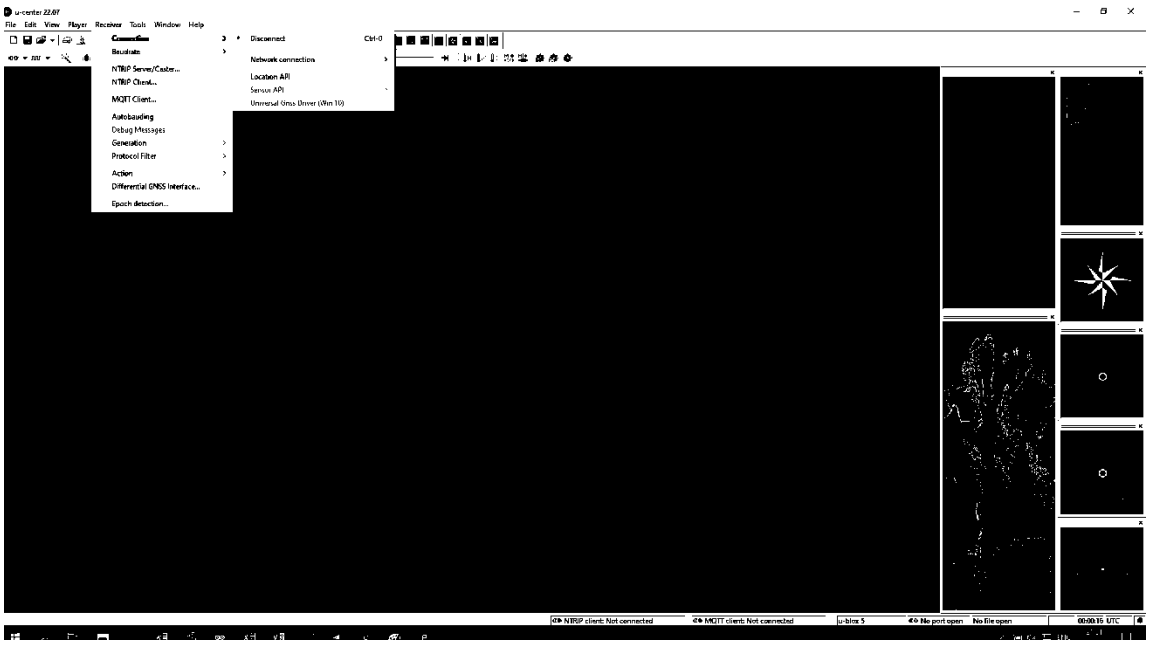


Рисунок 1.8 програма u-center

У цьому вікні буде показано довготу, широту, висоту та режим фіксації. Він також покаже HDOP, який є горизонтальним зниженням точності. Чим менше, тим краще, будь-що нижче 1,0 означає, що у вас хороший сигнал.

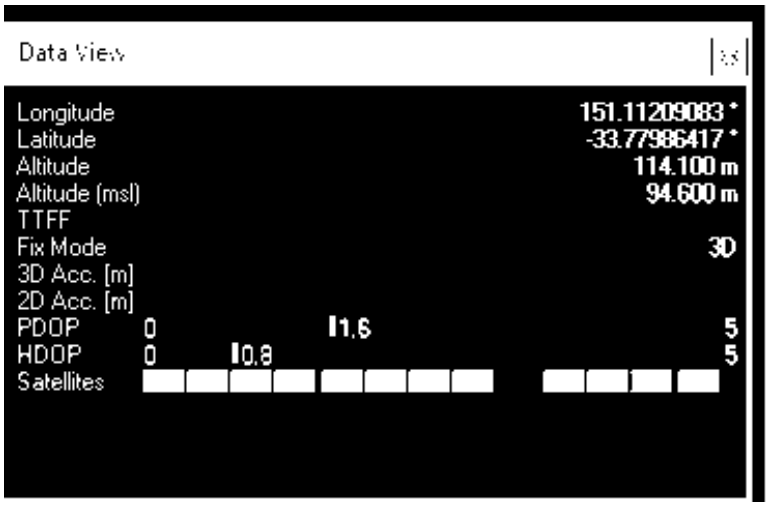


Рисунок 1.9 Вивід даних

Програмне забезпечення для оцінки стільникового зв’язку m-center — це потужний і простий у використанні інструмент для оцінювання, налаштування та тестування модулів стільникового зв’язку u-blox. Він містить інтуїтивно зрозумілий, простий для розуміння та використання графічний інтерфейс. м-центр працює безкоштовно.

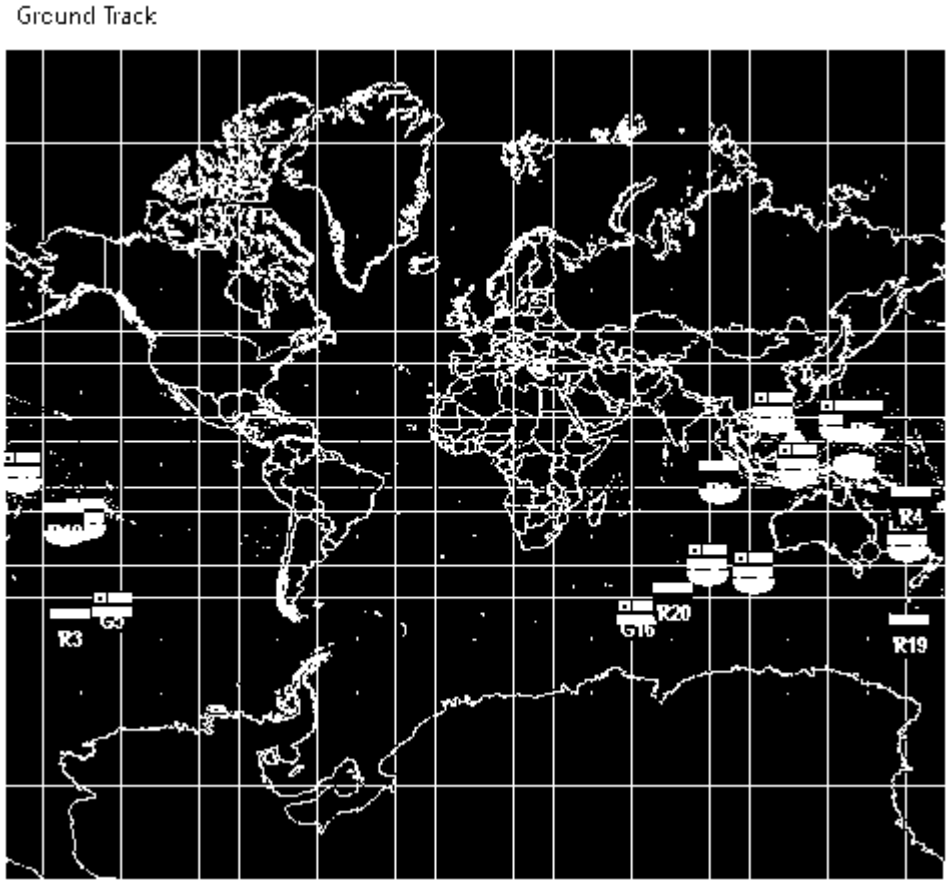


Рисунок 1.10 Розташування спутників

m-center надає зручні засоби для налаштування u-blox стільникових модулів і збережіть конфігурацію в модулі EPROM. Також можна переглядати та редагувати записи телефонної книги SIM, надсилати текстові повідомлення та спілкуватися зі стільниковим модулем за допомогою AT-команд. m-center можна використовувати для відновлення заводських налаштувань стільникового модуля та для виконання трасування. Крім того, при використанні комплектів оцінки стільникового зв'язку u-blox m-center забезпечує простий зв'язок із вбудованим модулем GNSS. m-center реалізує симулятор eCall IVS; Використовуючи його, клієнти можуть розпочати зв'язок eCall, визначаючи вміст даних MSD та правильно заповнюючи потрібні повідомлення.

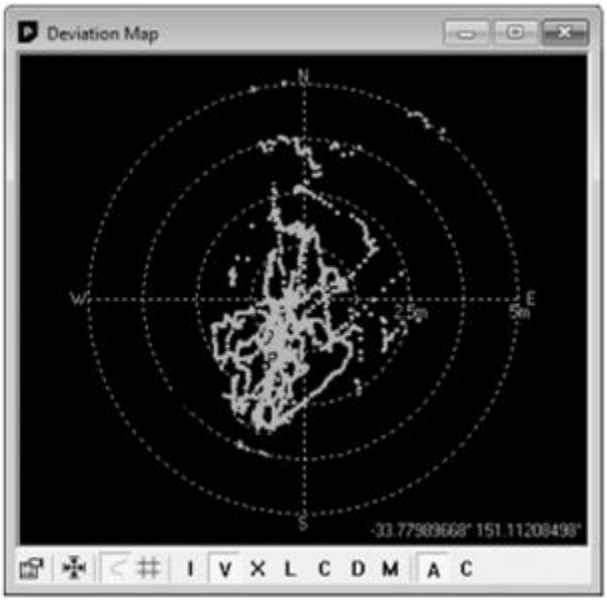


Рисунок 1.11 Карта відхилень

Sky view є чудовим інструментом для аналізу роботи антен, а також умов супутникового середовища спостереження.

Рисунок 1.12 Skye View

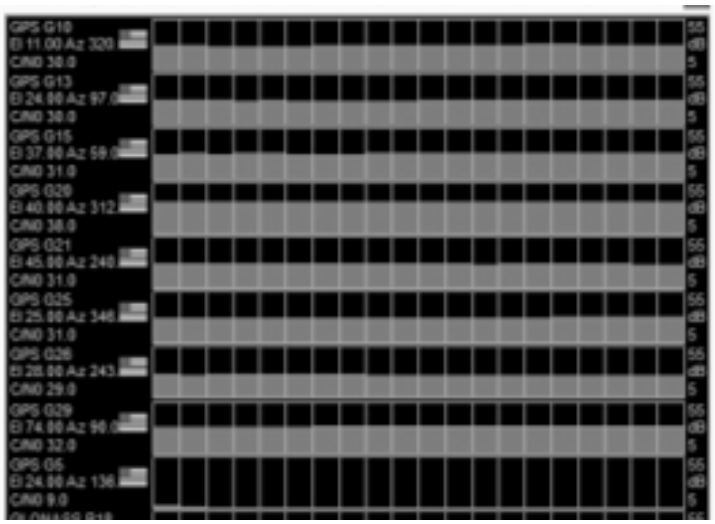


Рисунок 1.13 Історія супутникового сигналу

Дуже простий графічний інтерфейс користувача дозволяє користувачеві виконувати типові завдання LTE/UMTS/GPRS/GSM, а вбудований AT-термінал показує трасування всіх AT-команд, щоб зменшити час навчання набору AT-команд. Усю діяльність термінала AT можна зберегти в текстовий файл. Увімкнення функції трасування дозволяє користувачам зберігати внутрішні LTE/ Активність модуля UMTS/GPRS/GSM для полегшення надсилання двійкових даних до служби підтримки клієнтів u-blox.

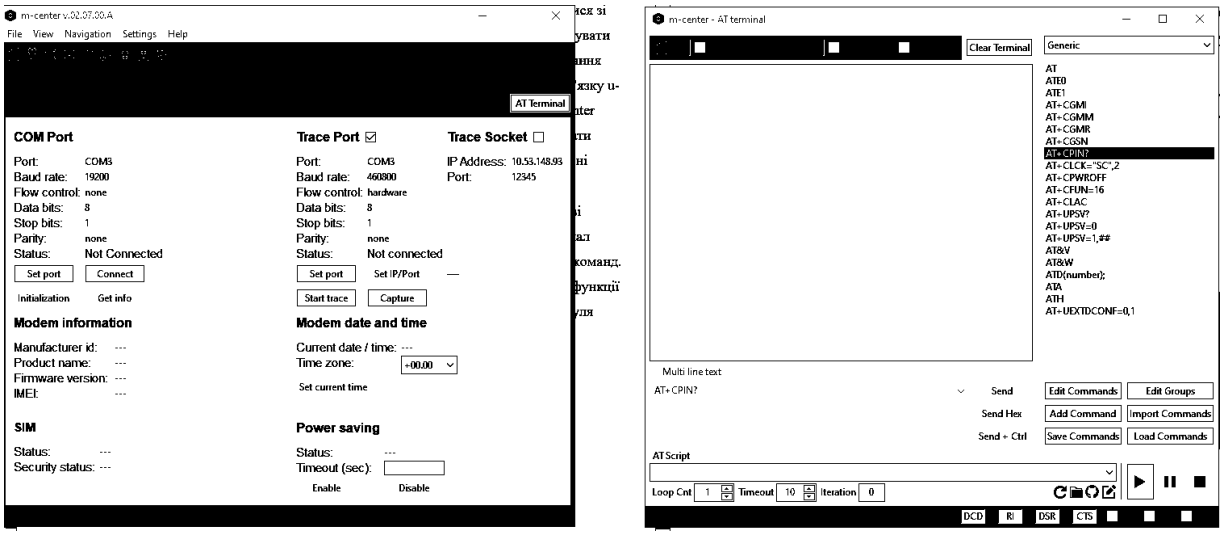


Рисунок 1.14 Програма m-center

GPS-відстеження та GSM-відстеження здійснюються приймачем, який збирає дані щонайменше з 4 супутників, щоб визначити точне місцезнаходження. Пристрої відстеження GPS GSM виконують це завдання, посилаючись на інформацію з вишки стільникового зв'язку, яка є найближчою до пристрою відстеження GSM/GPS. Серед цих двох технологій системи GPS здатні набагато точніше вимірювати місцезнаходження в межах одного метра, тоді як за допомогою технології відстеження GSM позиціонування можна визначити лише в межах 10 метрів. На сучасному ринку технологія GSM/GPS відстеження широко впроваджується в автомобілях із використанням GSM відстеження транспортних засобів, у стільникових телефонах, годинниках та в будь-якому іншому активі, який може бути зацікавлений у відстеженні. Завдяки таким маленьким пристроям, як пачка сірників, розмістити GPS/GSM-трекер тепер простіше, ніж будь-коли! Багато смартфонів, доступних сьогодні на ринку від таких виробників, як Blackberry, використовують декілька технологій для відстеження GSM (GPRS) / GPS. У цій ситуації модуль GPS у телефоні приймає супутникові сигнали, щоб визначити його точне місцезнаходження. Потім програмне забезпечення, встановлене на телефоні, перетворює інформацію GPS і використовує можливості телефону GPRS або GSM для передачі даних через стільникову мережу постачальника послуг. Останні досягнення в програмному забезпеченні, що підтримує відстеження транспортних засобів GSM, дозволили цій технології витіснити альтернативи GPS як найбільш

підходяще рішення для багатьох сценаріїв. Навіть у системах, які отримують дані про місцезнаходження за допомогою GPS, GSM-відстеження транспортних засобів дозволяє передавати потужну інформацію, таку як місцезнаходження, напрямок руху, швидкість, холостий хід двигуна, час запуску та зупинки двигуна тощо. Завдяки безлічі доступного надійного обладнання на ринку легко втратити з поля зору єдиний найважливіший аспект роботи будь-якої системи відстеження GPS або GSM для досягнення максимальної користі. Спосіб подання інформації зрештою визначить, наскільки ретельно ваш бізнес включає GPS або GSM трекер і використовує доступну інформацію.

2 ВИКОРИСТАННЯ МОВИ ПРОГРАМУВАННЯ PYTHON В СИСТЕМІ ВІДСТЖУВАННЯ КООРДИНАТ.

2.1 Мова програмування Python для IoT.

Для багатьох розробників Python легко засвоїти, він має чіткий синтаксис. Часто прототипи або реальні системи Інтернету речей (IoT) потрібно розробляти швидко та ефективно. Коли це відбувається, одразу постають два завдання: програмування пристроїв IoT та організація серверної частини, яка взаємодіє з цими пристроями. В обох завданнях ви можете використовувати Python як мову розробки. Або ви можете використовувати повністю функціональну і практичну версію MicroPython для роботи на пристроях з невеликими обчислювальними ресурсами, і відповідно, за дуже низькою ціною. Давайте подивимося, як можна використовувати Python для програмування пристроїв IoT і створити серверну частину для їх роботи. IoT з python став ефективним інструментом для прототипування, розробки та експлуатації різноманітних пристроїв і систем Інтернету речей. Python можна ефективно використовувати для програмування пристроїв IoT, а також для розробки відповідного серверу. Швидкість розробки, низький поріг для вивчення Python і великий набір бібліотек Python роблять цю мову програмування незамінною для IoT.

Для багатьох розробників Python є мовою вибору на ринку. Його легко освоїти, він має чистий синтаксис і підтримується великою онлайн-спільнотою. В IoT Python є чудовим вибором для серверної частини розробки, а також для

розробки програмного забезпечення пристроїв. Крім того, Python доступний для роботи на пристроях Linux, і ви можете використовувати MicroPython для мікроконтролерів. У цій статті сказано, що «використання цієї мови кодування [допомагає] зменшити обсяг даних, з якими вам доведеться мати справу і які доступні в хмарі. Незалежно від того, створюєте ви свій проект IoT з нуля чи взаємодієте з датчиками, приводами та аксесуарами, Python розпізнає ваші вимоги». Одними з багатьох переваг роботи з Python для пристроїв IoT є швидкість, з якою ви можете розробляти код, і велика кількість бібліотек для всіх видів платформ. Python є чудовим союзником для розробки прототипів пристроїв. Навіть якщо ви перепишете частину свого коду під час виробництва на C, C++ або Java, щоб покращити продуктивність, загалом система працюватиме ідеально на Python.

MicroPython - це реалізація мови програмування Python 3, яка включає основну частину підмножини стандартної бібліотеки Python і оптимізована для роботи на мікроконтролерах.

Існує кілька категорій модулів MicroPython:

- модулі не призначені для розширення користувачем;
- модулі з можливістю розширення користувачем (за допомогою коду Python);
- модулі, що реалізують розширення MicroPython до стандартних бібліотек Python;
- модулі, специфічні для певного порту MicroPython і, отже, не портативні.

Розвиток бездротового зв’язку та поява сенсорних технологій IoT тягне за собою появу абсолютно нових завдань, таких як:

- керуйте споживанням енергії.
- розробка стандартів зв'язку в мережі IoT.
- знизьте вартість ефективної інтеграції датчиків.

В MicroPython за основу взятий синтаксис Python якщо ви вже знайомі з Python у вас не буде проблеми. MicroPython чудово підійде якщо ви хочете зробити не великий DIY проект.

2.2 Фреймворки, бібліотеки Python для створення веб-додатка.

Paho-MQTT - це протокол, створений спеціально для Інтернету речей! Він головним чином зосереджений на високошвидкісному зв'язку з низьким навантаженням між пристроями, що надають ресурси. Він надає дуже зручну версію основних необхідних протоколів для використання у вбудованих системах. Запити MQTT не потребують зовнішнього налаштування; їх можна зробити безпосередньо в самому Python. Дуже зручно на етапі створення прототипу.

Tensorflow — це в основному чисельні обчислення, оскільки він побудований на Numpy. Він збирає інше математичне представлення, яке називається повільними графами даних, де вузли призначені для математичних операцій, а ребра — як масиви даних. Ця бібліотека є дуже корисним інструментом для роботи з великою кількістю нелінійних наборів даних.

Tkinter- хоча ця бібліотека попередньо встановлена в кожному варіанті Python, її надійність заслуговує на згадку. Tkinter — це бібліотека розробки GUI. Програмісти, які звикли до ООП, можуть спочатку здатися цій бібліотеці трохи лякаючою, але її винагорода з лишком компенсує зусилля. Ця бібліотека в основному використовується для тестування функціональності або багаторазового виконання того самого коду. Як у спідометрах, де один і той самий код потрібно запускати знову і знову, щоб тримати водія в курсі швидкості автомобіля, щоб уникнути будь-яких незручностей, які можуть призвести до катастрофи.

OpenCV - це дуже велика бібліотека з відкритим кодом машинного навчання та обробки зображень. Це дуже корисний інструмент, який переносить майбутнє програмування в повсякденне життя. Він може обробляти зображення, відео. При використанні з іншими бібліотеками, такими як NumPy, яка використовується для числових операцій, кількість функцій збільшується у вашому арсеналі, будь-які операції, які ви можете виконувати в NumPy, можна поєднувати з OpenCV. Це

найуспішніша бібліотека обробки зображень, оскільки вона містить високорівневі варіанти обробки зображень, що значно полегшує цей процес.

Pandas — ще один дуже корисний інструмент, призначений для науковців з даних. Будучи альтернативою MySQL, він дуже підходить для роботи з базами даних і обробки даних. Будучи новішою версією та створеною на Numpy, ця бібліотека є набагато оптимізованою та обробляє прямі операції з локальними наборами даних і здатність обробляти різнорідні та невпорядковані дані.

Matplitlib - ви можете використовувати цей фреймворк за допомогою команди: “import matplotlib. pyplot as plt” (Тепер пакет pyplot може називатися plt). Будучи розробником IoT, вам щосекунди доводиться мати справу з інтенсивним трафіком даних. Робота з великою кількістю інформації тепер увійшла у ваші вени. Візуалізація даних є найважливішою справою, з якою доводиться мати справу IoT або будь-якому Data Scientist. Це виглядає досить круто, якщо перетворити всі ці цифри на круті згинальні форми. Ця бібліотека є дуже корисним інструментом для перетворення ваших локальних даних на графіки та фігури.

Numpy коли ви глибше заглибитеся в масиви IoT, стане для вас дуже звичним. Після використання MatLab ви повинні бути добре знайомі з масивами та їх використанням. У Python масиви замінюються списками. Але ця заміна просто не підходить для кожного сценарію, і тут з’являється Numpy, пакет, який використовується для наукових обчислень через python (дуже схожий на MatLab, але легший). Найвидатнішою особливістю цієї бібліотеки є масове читання даних датчиків і робота з ними за допомогою вбудованих функцій.

Mysqldb база даних — це проста справа, коли йдеться про більшість додатків Інтернету речей. Якщо щось служить єдиній меті надсилання даних в Інтернет, база даних є обов’язковою вимогою, принаймні невелика, яка зберігає всю цю інформацію. І коли з’являється слово бази даних, MySQL є найпоширенішим рішенням, яке спадає на думку більшості розробників. У цій частині mysqldb є дуже зручним інструментом, який виконує команди в сценарії python для читання та запису в базу даних.

2.2.1 Опис використання Flask в проекті.

Мікрофреймворк Flask – це швидкий і безпроблемний інструмент для написання серверної частини для ваших систем IoT і легкого налаштування інформації вводу/виводу на стороні сервера, і він має багато функцій. Щоб почати, визначте запити, які вам потрібно обслуговувати з ваших пристроїв IoT, налаштуйте мікрофреймворк Flask і напишіть кілька рядків коду. Метод GET тепер повертатиме інформацію на запит клієнта. У багатьох випадках під час роботи з вашими пристроями IoT найкраще орієнтуватися на протокол RESTful. Це спрощує обмін між компонентами системи та дозволяє в майбутньому розширити систему обміну інформацією.

Недоліком використання цього підходу є потенційна відсутність ініціювання передачі даних із сервера на пристрій. Тобто IoT повинен самостійно і періодично знімати дані з сервера. Будьте спокійні, оскільки існують рішення для усунення цього ризику. Ви можете використовувати веб-сокети або бібліотеку Python для Pushsafer. За допомогою PushSafer ви можете легко та безпечно надсилати й отримувати push-сповіщення в режимі реального часу на свої пристрої iOS, Android і Windows (мобільні та настільні), а також у такі браузери, як Chrome, Firefox, Opera тощо.

WSGI (Web Server Gateway Interface) - це інтерфейс між веб-серверами та веб-програмами для обслуговування програми Flask.

SQLAlchemy - це інструментарій SQL і Object Relational Mapper. Він написаний на Python і надає повну потужність і гнучкість SQL для розробника додатків. SQLAlchemy славиться своїм об'єктно-реляційних відображенням (ORM), що дозволяє з самого початку чітко розв'язати об'єктну модель і схему бази даних. Flask-SQLAlchemy - це розширення для Flask, яке додає підтримку SQLAlchemy для додатка.

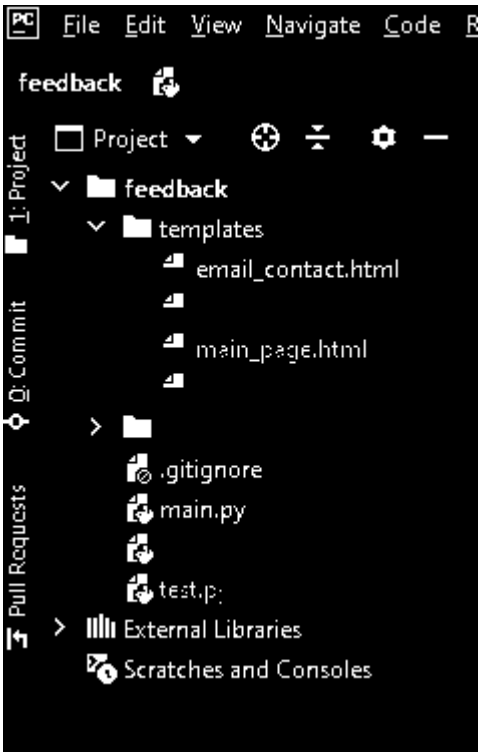


Рисунок 2.1 Базова структура додатка Flask

Ця структура має чотири папки верхнього рівня:

- додаток Flask міститься файл main.py;
- бази даних підключення через клієнт;
- модульні тести знаходяться в пакет тестів;
- папка venv містить віртуальне середовище Python де міститься інфорація про інтепретатор.

Також є кілька файлів:

- requirements.txt перераховує залежності бібліотек та фреймворків, щоб було легко відтворити ідентичне віртуальне середовище на іншому комп'ютері;
- confg.py зберігає налаштування конфігурації;
- flasky.py визначає екземпляр програми Flask, а також включає кілька завдань, які допомагають керувати додатком.

Програми Flask можна за бажанням виконувати в режимі налагодження. У цьому режимі за замовчуванням увімкнено два дуже зручні модулі сервера розробки — перезавантажувач і відладчик. Коли перезавантажувач увімкнено, Flask спостерігає за всіма файлами вихідного коду вашого проекту та автоматично

перезапускає сервер, коли будь-який із файлів змінюється. Запуск сервер з увімкненим перезавантажувачем надзвичайно корисний під час розробки, тому що кожного разу, коли ви змінюєте та зберігаєте вихідний файл, сервер автоматично перезапускається та підбирає зміни. Налагоджувач — це веб-інструмент, який з'являється у вашому браузері, коли ваша програма викликає необроблений виняток. Вікно веб-браузера перетворюється на інтерактивне трасування стека, яке дозволяє вам перевіряти вихідний код і оцінювати вирази в будь-якому місце в стеку викликів.

Обробка дат і часу у веб-додатку не є тривіальною проблемою, коли користувачі працюють у різних частинах світу. Сервер потребує єдиних одиниць часу, які не залежать від місця розташування кожного користувача, тому зазвичай використовується всесвітній координований час (UTC). Однак для користувачів перегляд часу, вираженого в UTC, може збити з пантелику, оскільки користувачі завжди очікують побачити дати та час, поданий за місцевим часом і відформатований відповідно до звичаїв свого регіону. Елегантне рішення, яке дозволяє серверу працювати виключно за UTC, полягає в надсиланні цих одиниць часу у веб-браузер, де вони перетворюються на місцевий час і відображаються за допомогою JavaScript. Веб-браузери можуть набагато краще виконувати це завдання, оскільки вони мають доступ до налаштувань часового поясу та мови на комп'ютері користувача.

після надсилання форми лише необроблений текст Markdown надсилається із запитом POST; попередній перегляд HTML, який відображається на сторінці, відхиляється. Надсилання згенерованого попереднього перегляду HTML разом із формою можна вважати загрозою безпеці, оскільки зловмиснику було б досить легко створити послідовності HTML, які не відповідають джерелу Mark-down, і надіслати їх. Щоб уникнути будь-яких ризиків, надсилається лише вихідний текст Markdown, який після потрапляння на сервер знову перетворюється на HTML за допомогою Markdown, конвертера Python Markdown-to-HTML. Отриманий HTML дезінфікується за допомогою Bleach, щоб забезпечити використання лише короткого списку дозволених тегів HTML. Перетворення дописів блогу Markdown

у HTML можна здійснити в шаблоні _posts.html, але це неефективно, оскільки дописи доведеться конвертувати щоразу, коли вони відображаються на сторінці. Щоб уникнути цього повторення, перетворення можна виконати один раз під час створення публікації в блозі, а потім збережено в базі даних. HTML-код для відтвореної публікації блогу кешується в новому полі, доданому до моделі публікації, що шаблон може отримати прямий доступ.

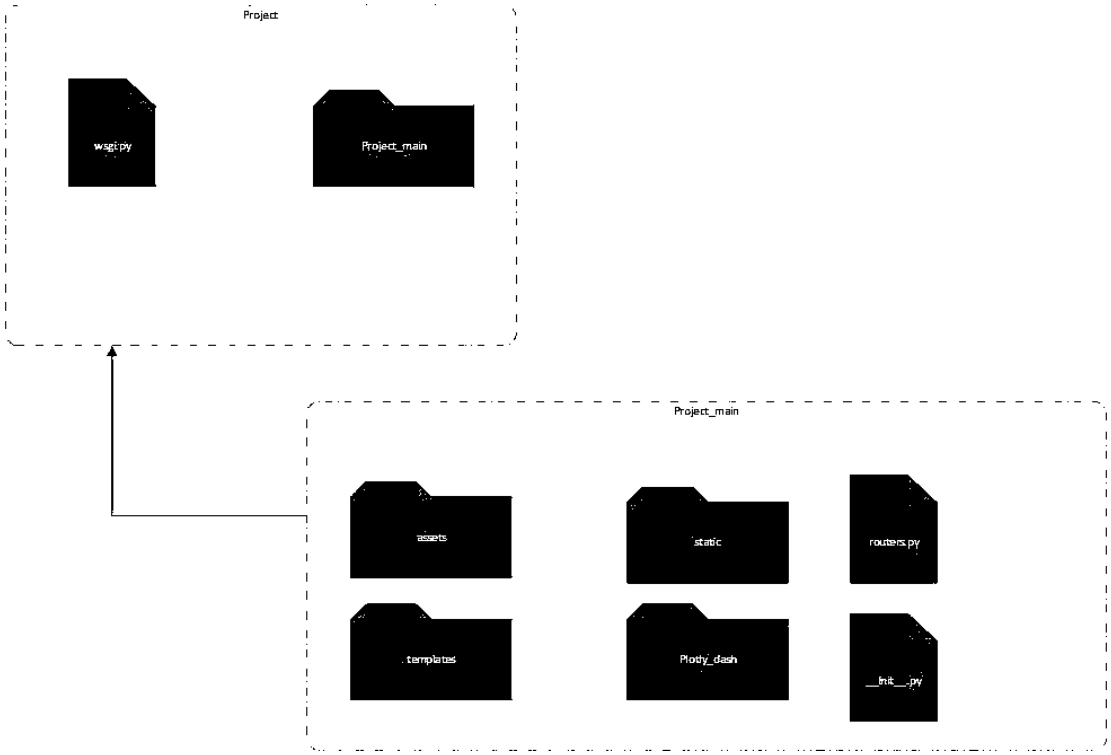


Рисунок 2.1 Структура додатка

2.2.2 Аналіз та візуалізація за допомогою бібліотеки gmaps.

Для більшості операцій на Картах Google ви повинні повідомити Google, хто ви. Щоб пройти автентифікацію за допомогою Google Maps, вам потрібно перейти в Google Cloud Console і створіть новий проект, а потім створіть свій ключ API з облікових даних. Ключ API - це рядок, який починається з літер AI.

Також після створення ключа API вам потрібно ввімкнути Maps Javascript API.

Це дає вам повноцінну карту Google. Ви можете збільшувати та зменшувати масштаб, перемикатися на вигляд із супутника та навіть на перегляд вулиць, якщо дійсно хочете. Теплова карта змінюється під час збільшення та зменшення

масштабу. Масив locations може бути або списком кортежів, як у прикладі вище, масивом numpy форми $N, помножених на 2$, або фреймом даних із двома стовпцями. Більшість атрибутів на базовій карті та шарах можна встановити за допомогою іменованих аргументів у конструктора або як атрибути екземпляра після створення екземпляра.

Рівень малювання дозволяє вказати код Python, який буде виконуватись щоразу, коли користувач додає об’єкт (наприклад, маркер, лінію чи багатокутник) на карту. Щоб продемонструвати це, ми створимо невелику програму для зворотного геокодування: коли користувач розміщує маркер на карті, ми знаходимо адресу, найближчу до цього маркера, і записуємо її в текстовий віджет. Ми будемо використовувати geopy, обгортку навколо кількох API геокодування, щоб обчислити адресу за координатами маркера.

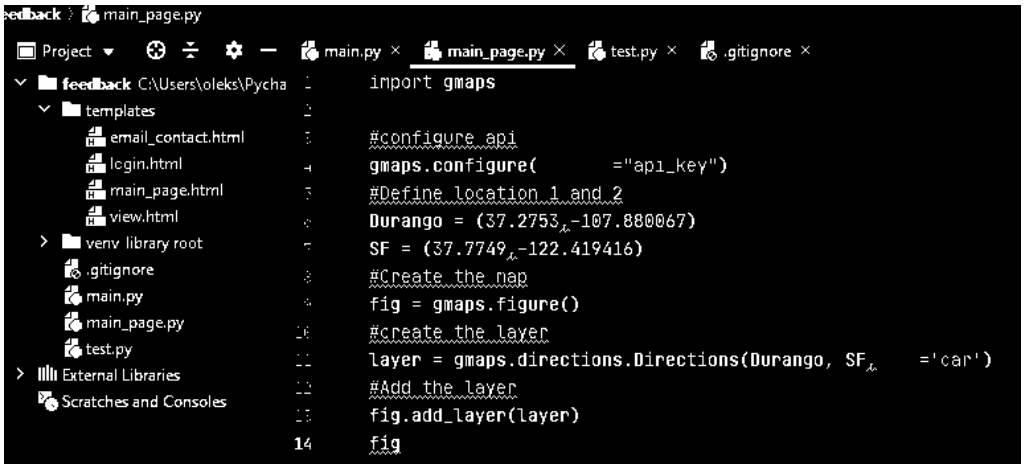


Рисунок 2.2 Пориклад використання gmaps

Для використання даної бібліотеки потрібно згенерувати ключ для доступу до ресурсів гугл, а саме карт для відображення.

OpenStreetMap створений спільнотою картографів, які додають і підтримують дані про дороги, стежки, кафе, вокзали та багато інших об'єктів по всьому світу. OpenStreetMap надає особливого значення знання місцевості. Учасники також використовують аерофотознімки, GPS-пристрої та низькотехнологічні карти земель для перевірки того, що дані OSM є точними та актуальними.

OpenStreetMap має API для редагування для отримання та збереження необроблених геоданих з/в базу даних OpenStreetMap — це вхідна сторінка для документації. Якщо ви просто хочете вставити карту у веб-сторінку, вам не потрібен цей API. Натомість використовуйте бібліотеку веб-карт. Як альтернативу, розгляньте Overpass API, який надає доступ до API лише для читання.

2.2.3 База даних MongoDB.

MongoDB — це система керування базами даних, призначена для швидкої розробки веб-додатків та інтернет-інфраструктури. Модель даних і стратегії збереження створені для високої пропускної здатності читання та запису та можливості легкого масштабування за допомогою автоматичного перемикання після відмови. Незалежно від того, чи потрібен програмі лише один вузол бази даних чи десятки, MongoDB може забезпечити напрочуд хорошу продуктивність. Якщо у вас виникли труднощі з масштабуванням реляційних баз даних, це може бути чудовою новиною. Але не всім це потрібно працювати в масштабі. Можливо, все, що вам коли-небудь було потрібно, це один сервер бази даних.

Формат документа MongoDB базується на JSON, популярній схемі для зберігання довільних структур даних. JSON — це абревіатура від JavaScript Object Notation. Як ви щойно бачили, Структури JSON складаються з ключів і значень, і вони можуть вкладатися довільно глибоко. Вони аналогічні словникам і хеш-картам інших мов програмування. Модель даних на основі документів може представляти багаті ієрархічні структури даних. Часто можна обійтися без багатотабличних об’єднань, поширених у реляційних базах даних. Наприклад, припустімо, що ви моделюєте продукти для сайту електронної комерції. За допомогою моделі документа, навпаки, більшу частину інформації про продукт можна представити в одному документі. Коли ви відкриваєте оболонку MongoDB JavaScript, Ви можете легко отримати зрозуміле представлення свого продукту з усією його інформацією, ієрархічно організованою у структурі, подібній до JSON. Ви також можете запитувати його та маніпулювати ним. Можливості запитів MongoDB розроблені спеціально для роботи зі структурованими документами,

тому користувачі, які переходять з реляційних баз даних, відчувають аналогічний рівень потужності запитів. Крім того, більшість розробників зараз працюють з об'єктно-орієнтованим мовами, і вони хочуть сховище даних, яке краще відображає об’єкти. За допомогою MongoDB об’єкт, визначений у мові програмування, часто можна зберегти як є, усуваючи частину складності відображувачів об’єктів. Якщо ви маєте досвід роботи з реляційними базами даних, може бути корисним підійти до MongoDB з точки зору перенесення ваших наявних навичок у цю нову базу даних. На думку, це інструмент у наборі інструментів розробника, як і будь-яка інша база даних, і ви повинні знати його обмеження та сильні сторони. Деякі робочі навантаження вимагають реляційних з’єднань і іншого керування пам’яттю, ніж MongoDB. З іншого боку, документна модель особливо добре підходить для деяких робочих навантажень, а відсутність схеми означає, що MongoDB може бути одним із найкращих інструментів для швидкої розробки та ітерації програми. Наша мета — надати вам інформацію, яка вам потрібна, щоб вирішити, чи підходить вам MongoDB, і пояснити, як її ефективно використовувати.

Проектування схеми бази даних — це процес вибору найкращого представлення для набору даних, враховуючи особливості системи бази даних, характер даних і вимоги програми. Принципи проектування схем для систем реляційних баз даних добре встановлені. За допомогою RDBMS вам рекомендується шукати нормалізовану модель даних, яка допомагає забезпечити загальну можливість запиту та уникнути оновлень даних, які можуть призвести до невідповідностей. Крім того, встановлені шаблони не дозволяють розробникам задаватися питанням, як змоделювати, скажімо, зв’язки «один до багатьох» і «багато до багатьох». Але проектування схем ніколи не є точною наукою, навіть з реляційними базами даних. Функціональність і продуктивність додатків є найвищим майстром у розробці схем, тому кожне «правило» має винятки. Якщо ви зі світу RDBMS, вас може турбувати відсутність у MongoDB жорстких правил розробки схем. Хороші практики з’явилися, але все ще зазвичай існує більше ніж один хороший спосіб моделювання певного набору даних.

Усі документи серіалізуються в BSON перед надсиланням до MongoDB, пізніше вони десеріалізуються з BSON. Драйвер обробляє цей процес і перекладає його з і на відповідні типи даних у своїй мові програмування. Більшість драйверів забезпечують простий інтерфейс для серіалізації з і до BSON, це відбувається автоматично під час читання та запису документів.

MongoDB не використовує SQL. Натомість він має власну мову запитів, схожу на JSON. Ви досліджували цю мову протягом усієї книги, але тепер давайте повернемося до більш ємких прикладів із реального світу. Мова запитів MongoDB в цілому, детально розглядаючи кожен доступний оператор запиту. У поточному стані оволодіння запитами та агрегаціями в MongoDB — це не стільки питання картографування кожного куточка, скільки пошук найкращих способів виконання повсякденних завдань.

Структура агрегації — це розширена мова запитів MongoDB, яка дозволяє трансформувати та поєднувати дані з кількох документів, щоб генерувати нову інформацію, недоступну в жодному окремому документі. Наприклад, ви можете використовувати структуру агрегації, щоб визначити продажі за місяцями, продажі за продуктами або загальну кількість замовлень за користувачами. Для тих, хто знайомий з реляційними базами даних, ви можете розглядати структуру агрегації як еквівалент MongoDB пропозиції SQL GROUP BY. Хоча ви могли обчислити цю інформацію раніше за допомогою можливостей зменшення карти MongoDB або в програмному коді, структура агрегації робить це завдання набагато простішим і ефективнішим, дозволяючи визначати серію операцій з документами, а потім надсилати їх у вигляді масиву до MongoDB за один виклик.

MySQL — це база даних на основі SQL, яка зберігає дані в таблицях зі стовпцями та рядками та працює лише зі структурованими даними. MongoDB, з іншого боку, може обробляти неструктуровані дані та зберігати JSON-подібні документи, а не таблиці, і використовує мову запитів MongoDB для зв’язку з БД. MySQL є надзвичайно відомим база даних із величезною спільнотою та чудовою стабільністю, і MongoDB є такою це досить нова технологія, спільнота якої зростає та розроблена MongoDB Inc. MySQL є вертикально масштабованим, у якому

навантаження на один сервер можна збільшити шляхом оновлення оперативної пам’яті, SSD або процесора, тоді як у випадку MongoDB, який є горизонтально масштабованим, йому потрібно спільно використовувати та додавати більше серверів, щоб збільшити навантаження на сервер. MongoDB є кращим вибором для великих навантажень на запис і великих наборів даних, а MySQL ідеально підходить для додатків, які сильно залежать від багаторядкових транзакцій, таких як системи обліку. MongoDB є чудовим вибором для додатків із динамічною структурою та високим навантаженням на дані, таких як аналітичні додатки в реальному часі або системи керування вмістом. Flask підтримує взаємодію з MySQL і MongoDB. Існують різноманітні власні драйвери, а також ORM/ODM для зв’язку з базою даних. Flask-MySQL — це розширення Flask, яке дозволяє власне підключення до MySQL; Flask-PyMongo — це рідне розширення для роботи з MongoDB у Flask, яке також рекомендовано MongoDB. Flask-MongoEngine — це розширення Flask, ODM для Flask для роботи з MongoDB. Flask SQLAlchemy — це рівень ORM для додатків Flask для підключення до MySQL.

MongoDB Compass — це фантастичний інструмент для тих, хто не знає, як використовувати запити командного рядка для інтерпретації та обробки даних. MongoDB Compass можна легко завантажити, а потім інсталювати в Windows, виконавши кілька простих кроків. Це нескладний процес, який займає мінімум часу. Це потужний графічний інтерфейс користувача, який дає змогу зручно взаємодіяти з системою баз даних MongoDB. Це вигідно для вас, оскільки не вимагає від вас жодної технічної підготовки чи попередніх знань із синтаксису запитів MongoDB. Однак MongoDB Compass можна використовувати не тільки для відображення даних; його також можна використовувати для оптимізації запитів, керування індексами та перевірки документів.

MongoDB Compass популярний завдяки таким унікальним функціям:

- Легке надсилання запитів: Оскільки попереднього розуміння запитів MongoDB не потрібно, це робить надсилання запитів до бази даних MongoDB надзвичайно простим.

- Візуалізація індексів: це дає вам змогу мати повну візуалізацію для всіх індексів, присутніх у MongoDB.
- Ефективне створення правил: користувачі мають повний контроль над правилами перевірки схеми під час створення документів.
- Конструктор конвеєрів агрегації: ця функція дозволяє агрегувати дані в набори шляхом обробки колекції документів або представлень на кількох рівнях.
- Аналіз схем і структури. Схеми — це чудовий спосіб упорядкувати дані у форматі JSON, оскільки вони надають вам низку різних методів, які можна застосувати до даних і отримати дуже конкретні результати. Це ще більше прискорює процес аналізу даних.

Щоб отримати дані в об’єктну форму для використання в програмах, більшість баз даних вимагають використання складних оболонок, таких як ORM (Object Relational Mappers). Рішення MongoDB зберігати та представляти дані у форматі документа означає, що ви можете отримати до них доступ будь-якою мовою, використовуючи власні структури даних.

Якщо ви звикли перевести свій сайт або програму в автономний режим, щоб змінити формат даних, MongoDB створено для змін. MongoDB докладає багато зусиль для створення ефективних методів і навчання на наших невдачах, але база даних зазвичай є обмеженням. Зміна схем є простим і швидким процесом, і ви також можете додавати нові дані до MongoDB у будь-який час, не викликаючи перерв у її процесах. MongoDB створено, щоб полегшити пошук даних і рідко потребує об’єднань або транзакцій, але вона більш ніж здатна обробляти складні запити. За допомогою лише кількох рядків JSON-подібного MQL ви можете робити глибокі запити в документах і навіть запускати складні конвеєри аналітики.

2.3 Передача даних використовуючи протокол MQTT.

Технологія IBM Websphere Message Queue була вперше вигадана у 1993 р. для розв'язання проблем у незалежних та неконкурентних розподілених системах для забезпечення захищеного зв'язку. Видання-підписка, також відома як pub/sub,

є способом відокремити клієнта, що передає повідомлення від іншого клієнта, що отримує повідомлення. На відміну від традиційної моделі клієнт-сервер, клієнти не обізнані про будь-які фізичні ідентифікатори, на зразок IP-адреси або порту. MQTT – це архітектура pub/sub, але не черга повідомлень. Черги повідомлень за своєю природою зберігають повідомлення, а MQTT – ні. У MQTT якщо ніхто не підписується (або не слухає) на тему, вона просто ігнорується і втрачається. Черги повідомлень також підтримують топологію клієнт-сервер, де один споживач з'єднаний з одним виробником.

Посередника можна налаштувати за допомогою файлу конфігурації, як описано в mosquitto.conf, і це основна інформація для mosquitto. Файли, необхідні для підтримки SSL/TLS, описані в mosquitto-tls.

Деякі версії Windows мають обмеження на кількість одночасних підключень через використання Windows API. У сучасних версіях Windows, напр. Windows 10 або Windows Server 2019, це приблизно 8192 підключення. У попередніх версіях Windows це обмеження становить 2048 підключень. Mosquitto забезпечує повну підтримку MQTT v5.0, але деякі функції не використовуються безпосередньо. У наступних розділах описано нові функції та пояснено, де Mosquitto не використовує цю функцію. Базова автентифікація MQTT використовує перевірку імені користувача та пароля. Розширена автентифікація дозволяє інтегрувати в MQTT різні схеми автентифікації, і навіть схеми з кількома кроками. Клієнти запитують певний тип автентифікації, і якщо брокер налаштовано для цієї схеми, автентифікація продовжується. Mosquitto підтримує розширену автентифікацію за допомогою плагінів. Більшість пакетів MQTT тепер мають концепцію коду причини, який вказує на успішне або невдале виконання, а також на причину невдачі. Mosquitto забезпечує повну підтримку кодів причин, але не використовує функцію рядка причини, яка може бути використана для надання зрозумілого людині рядка помилки для пояснення коду причини.

MQTT успішно відокремлює видавців від споживачів. Оскільки брокер є керівним органом між видавцями та споживачами, нема потреби безпосередньо ідентифікувати видавця та споживача на основі фізичних даних (таких як IP-

адреса). Керовані хмарно брокери MQTT зазвичай можуть поглинати мільйони повідомлень на годину та підтримувати десятки тисяч видавців. MQTT не залежить від формату даних. Корисне навантаження може містити будь-який тип даних, тому і видавці, і передплатники повинні розуміти та узгоджувати формат даних. Можна надсилати текстові повідомлення, дані зображення, аудіодані, зашифровані дані, двійкові дані, об'єкти JSON або будь-яку іншу структуру в корисному навантаженні. Проте текстові та двійкові дані JSON є найпоширенішими типами даних корисного навантаження.

Незважаючи на те, що MQTT заснований на TCP, з'єднання все ще можуть обриватися, особливо у випадку бездротових датчиків. Пристрій може втратити живлення, втратити сильний сигнал або може бути просто польова поломка, і сеанс перейде у напіввідкритий стан. Тоді сервер буде вважати, що з'єднання, як і раніше, надійне і очікувати дані. Щоб вийти із цього напіввідкритого стану, MQTT використовує систему keep-alive. Використовуючи цю систему як брокер MQTT, так і клієнт мають гарантію того, що з'єднання залишається працездатним, навіть якщо протягом деякого часу не було передачі. Клієнт відправляє пакет PINGREQ брокеру, який, своєю чергою, підтверджує повідомлення за допомогою PINGRESP. Таймер встановлений на стороні клієнта та брокера. Якщо повідомлення не було передано ні ким із них протягом заданого проміжку часу, повинен бути надісланий пакет keepalive. Як PINGREQ, так і повідомлення, скинуть таймер keep-alive. Якщо keep-alive не отримано і час таймера закінчується, брокер закриє з'єднання і відправить LWT-пакет всім клієнтам. Клієнт може в якийсь момент пізніше спробувати знову під'єднатися. У цьому випадку брокер закриває напіввідкрите з'єднання та відкриває нове з'єднання з клієнтом.

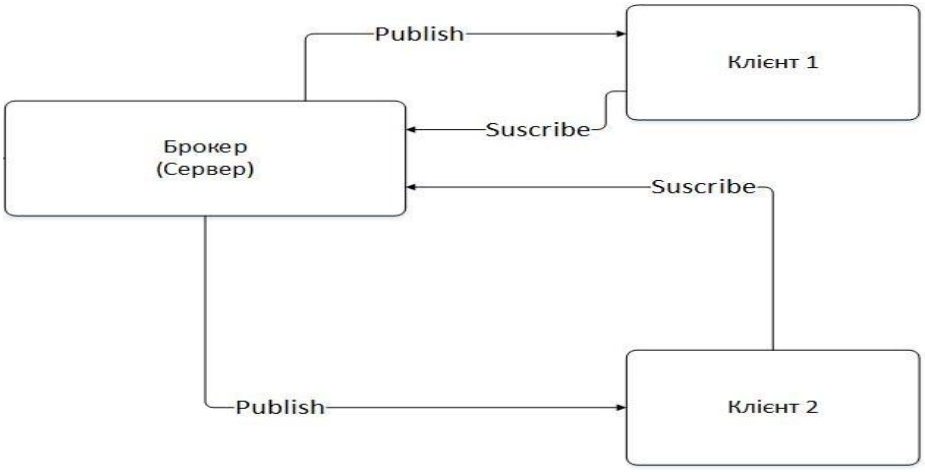


Рисунок 2.3 Протокол MQTT

Пакет MQTT знаходиться зверху рівня TCP мережевого стека моделі OSI. Пакет складається з 2-байтового фіксованого заголовка, який завжди повинен бути присутнім, заголовка зі змінним розміром (необов'язково) і закінчується корисним навантаженням.

Google унікальний тим, що для шифрування всіх пакетів даних із використанням JSON Web Tokens (JWT) та агента сертифіката потрібне сильне шифрування (TLS) поверх MQTT.

Веб-токен JSON — це відкритий галузевий стандарт, який використовується для обміну інформацією між двома об’єктами, як правило, клієнтом (наприклад, зовнішнім інтерфейсом програми) і сервером (сервером програми). Вони містять об’єкти JSON, які містять інформацію, якою потрібно поділитися. Кожен JWT також підписується за допомогою криптографії (хешування), щоб гарантувати, що вміст JSON (також відомий як претензії JWT) не може бути змінений клієнтом або зловмисною стороною. Вам може бути цікаво, чому сервер автентифікації не може просто надіслати інформацію як звичайний об’єкт JSON і чому йому потрібно перетворити її на «токен». Якщо сервер автентифікації надсилає його як звичайний JSON, API клієнтської програми не матимуть можливості перевірити, чи вміст, який вони отримують, правильний. Зловмисник може, наприклад, змінити ідентифікатор користувача (під заявка у наведеному вище прикладі JSON), і API

програми не зможуть дізнатися, що це сталося. Через цю проблему безпеки сервер автентифікації повинен передавати цю інформацію таким чином, щоб її могла перевірити клієнтська програма, і саме тут з'являється концепція «токена».

Простіше кажучи, токен — це рядок, який містить деяку інформацію, яку можна безпечно перевірити. Це може бути випадковий набір буквено-цифрових символів, які вказують на ідентифікатор у базі даних, або це може бути закодований JSON, який клієнт може самостійно перевірити (відомий як JWT). Можливо, ви помітили, що у прикладі JWT (випущеному Google) вище корисне навантаження JSON має неочевидні назви полів. Вони використовують sub, iat, aud тощо:

- iss: Емітент токена (у цьому випадку Google).
- azp і aud: ідентифікатори клієнта, видані Google для вашої програми. Таким чином Google дізнається, який веб-сайт намагається використовувати його службу входу, а веб-сайт знає, що JWT було видано спеціально для нього.
- sub: ідентифікатор користувача Google кінцевого користувача.
- at_hash: хеш маркера доступу. Маркер доступу OAuth відрізняється від JWT у тому сенсі, що він є непрозорим маркером. Мета маркера доступу полягає в тому, щоб клієнтська програма могла надсилати Google запит на отримання додаткової інформації про користувача, який увійшов у систему.
- email: ідентифікатор електронної пошти кінцевого користувача.
- email_verified: чи підтвердив користувач свою електронну адресу.
- iat: час (у мілісекундах з епохи), коли було створено JWT.
- exp: час (у мілісекундах з епохи), коли було створено JWT.
- nonce: може використовуватися клієнтською програмою для запобігання атакам відтворення.
- hd: розміщений домен G Suite користувача причиною використання цих спеціальних ключів є дотримання галузевих умов щодо назв

важливих полів у JWT. Дотримання цієї угоди дозволяє клієнтським бібліотекам різними мовами перевіряти дійсність JWT, виданих будь-якими серверами авторизації. Наприклад, якщо клієнтській бібліотеці потрібно перевірити, чи закінчився термін дії JWT, вона просто шукатиме поле iat.

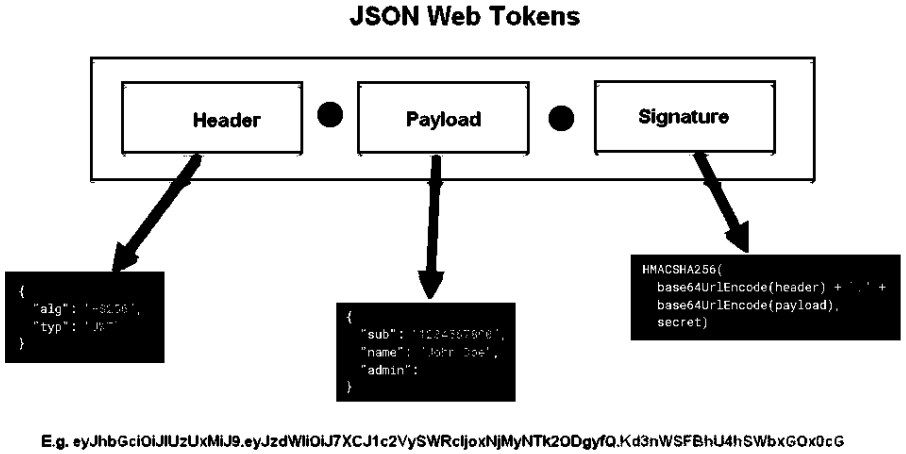


Рисунок 2.4 Приклад підписки на topic

Кожен пристрій створить кілька відкритих/закритих ключів. Google гарантує, що кожен пристрій має унікальний ідентифікатор та ключ у разі злому це торкнеться лише одного вузол і вкаже на загальну поверхню атаки.

Назви тем створюються:

- з урахуванням регістру;
- використовувати UTF-8;
- має складатися щонайменше з одного символу, щоб бути дійсним.

Рисунок 2.5 Приклад підписки на topic

MQTT містить три рівні якості обслуговування.

QoS-0 (незавірена передача) - це мінімальний рівень QoS. Це процес доставлення без підтвердження одержувачем повідомлення і без повторної передачі повідомлення відправником.

QoS-1 (гарантована передача) - цей режим гарантує доставлення повідомлення хоча б один раз. Повідомлення може бути доставлено декілька разів, і одержувач відправить назад підтвердження з відповіддю PUBACK.

QoS-2 (гарантований сервіс для додатків) - це найвищий рівень QoS, який переконується в доставлення й інформує відправника та одержувача, що повідомлення було передано вірно. PUBREL дозволяє одержувачеві безпечно відкинути будь-які повторні передачі цього повідомлення.

Мережа датчиків від MQTT називається MQTT-SN. Для MQTT-SN не потрібно стек TCP/IP. Його можна використовувати по послідовному з'єднанню. Він може використовуватися протокол UDP, який вимагає менше ресурсів, ніж TCP.

Різниця між MQTT й MQTT-SN.

- в MQTT-SN є три повідомлення CONNECT, а в MQTT – одне;
- попередньо визначені ідентифікатори тем можуть використовуватися без будь-якої реєстрації. Для використання клієнт і сервер повинні використовувати один і той же ідентифікатор теми;
- процедура виявлення, яка дозволяє клієнтам знаходити мережеві адреси серверів і шлюзів;
- в MQTT-SN використовується переглянута процедура keepAlive. Це робиться для підтримки сплячих клієнтів, в яких всі призначені для них повідомлення зберігаються сервером і передаються при пробудженні.

AMQP означає AdvancedAMQP означає Advanced Message Queuing Protocol. Це стабільний і вірний протокол MOM, що використовується такими великими джерелами даних, як JPMorgan Chase, для обробки більше 1 мільярда повідомлень на день і Ocean Observatory Initiative для збору більше 8 терабайт океанографічних даних щодня Message Queuing Protocol. Це стабільний і перевірений протокол MOM, що використовується такими великими джерелами даних, як JPMorgan Chase, для обробки понад 1 мільярд повідомлень на день і Ocean Observatory Initiative для збору понад 8 терабайт океанографічних даних щодня. AMQP –

система зв'язку, орієнтоване повідомлення і контроль потоків. Це протокол рівня проводів та низькорівневий інтерфейс. Дротовий протокол звертається до API відразу над фізичним рівнем мережі. API-інтерфейс на рівні каналу дозволяє різним службам обміну повідомленнями, таким як .NET (NMS) та Java (JMS), взаємодіяти один з одним. Аналогічно AMQP намагається відокремити видавців від передплатників. На відміну від MQTT, він має механізми для балансування навантаження та формалізації черги. Широко використовується протокол, заснований на AMQP, є RabbitMQ. RabbitMQ є брокером повідомлень AMQP, написаним на Erlang. Крім того, є кілька клієнтів AMQP, таких як клієнт RabbitMQ, написаний на Java, C#, Javascript і Erlang, а також Apache Qpid, написаний на Python, C++, C#, Java і Ruby. Один або кілька віртуальних хостів з власними просторами імен, обмінами та чергами повідомлень будуть на центральному сервері (серверах). Виробники та споживачі підписуються на послуги обміну. Служба обміну отримує повідомлення від видавця та надсилає дані у відповідну чергу. Топологія мережі AMQP заснована на принципі «зібери та роздай» і передбачає можливість взаємодії концентраторів між собою. AMQP складається з вузлів та зв'язків. Вузол є іменованим джерелом чи стоком повідомлень. Кадр повідомлення переміщається між вузлами за однонаправленими посиланнями. Якщо повідомлення надсилається через вузол, глобальний ідентифікатор не змінюється. Якщо вузол виконує будь-яке перетворення, призначається новий ідентифікатор.

3 РЕАЛІЗАЦІЯ MVP СИСТЕМИ «СИСТЕМИ ВІДСТЕЖЕННЯ КООРДИНАТ ОБ'ЯКТІВ НА ОСНОВІ МОДУЛІВ GSM, GPRS».

3.1 Структура веб-додатка.

Мінімальний життєздатний продукт містить в собі найменше, що ви можете створити, що забезпечує цінність та функціональність для споживача.

Кроки для створення мінімального життєздатного продукту:

- запуск відкритої бета-версії;
- збирати дані про використання та розпочати відстеження найважливіших показників;

- помилки та непотрібні функції;
- вбудовування змін у наступну версію продукту.

Веб-додаток містити зручний функціонал та зрозумілий дизайн. Додаток повинен відкриватись коректно на будь-яких пристроях які мають вихід в інтернет. Реалізація даного проекту буде здійснена з застосуванням Python, а саме фреймворк Flask для Back-end. Front-end буде використовувати HTML, CSS. Фреймворк gmaps відповідає за візуалізацію даних. База даних буде використовувати MongoDB для запису даних та выдображення у веб-форму.

Щоб користуватись додатком потрібно авторизуватись в систему через логін та пароль. При авторизації в систему буде відображено карту з мітками де знаходиться об'єкт та дані в табличній формі.

Веб-додаток забезпечує наступні функціональні можливості:

- легкість у використані;
- система сповіщення через пошту якщо відсутня відповідь від модуля;
- моніторин координат;

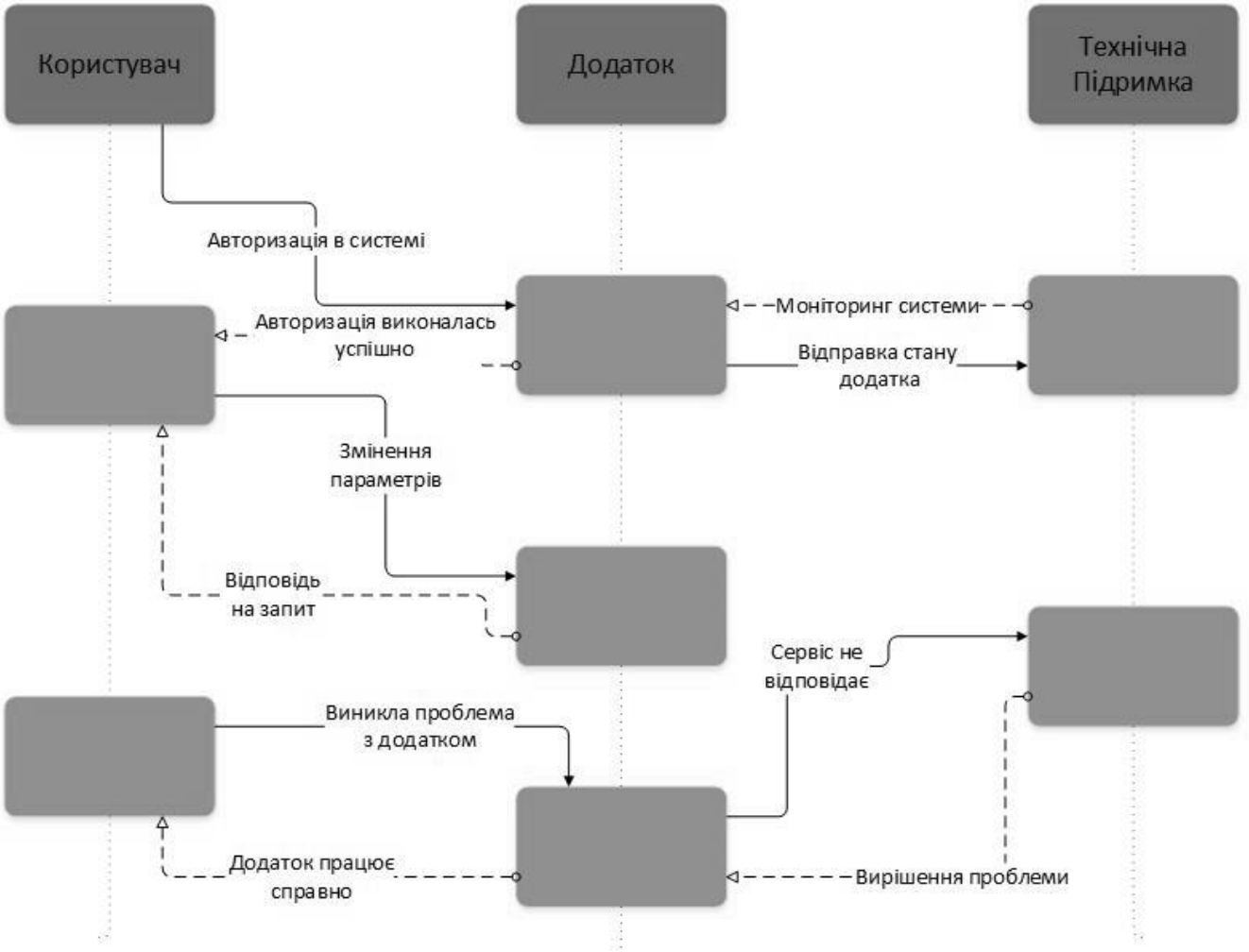


Рисунок 3.1 Діаграма зміна параметрів системі

Відображення взаємодії через додаток на стороні клієнта. Це використання, відоме як схема послідовності дії людей, які безпосередньо беруть участь у використанні системи. Він показує, як працює система, слідуючи процесу від початку до кінця.

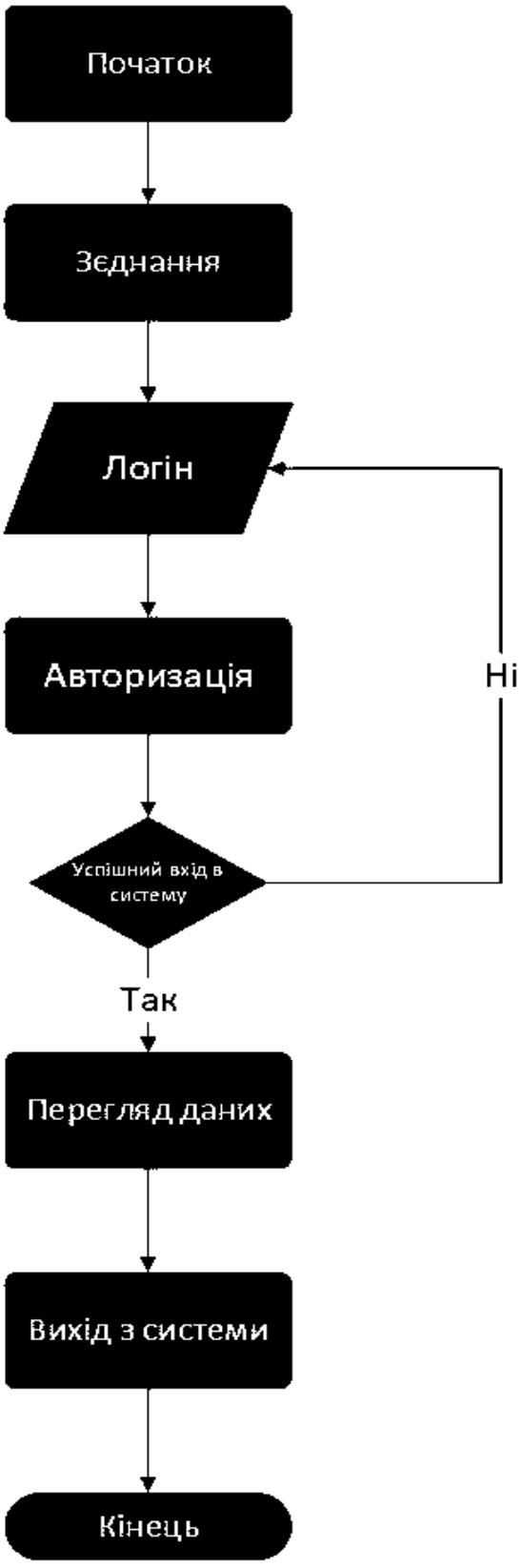


Рисунок 3.2 Схема взаємодії на стороні клієнта

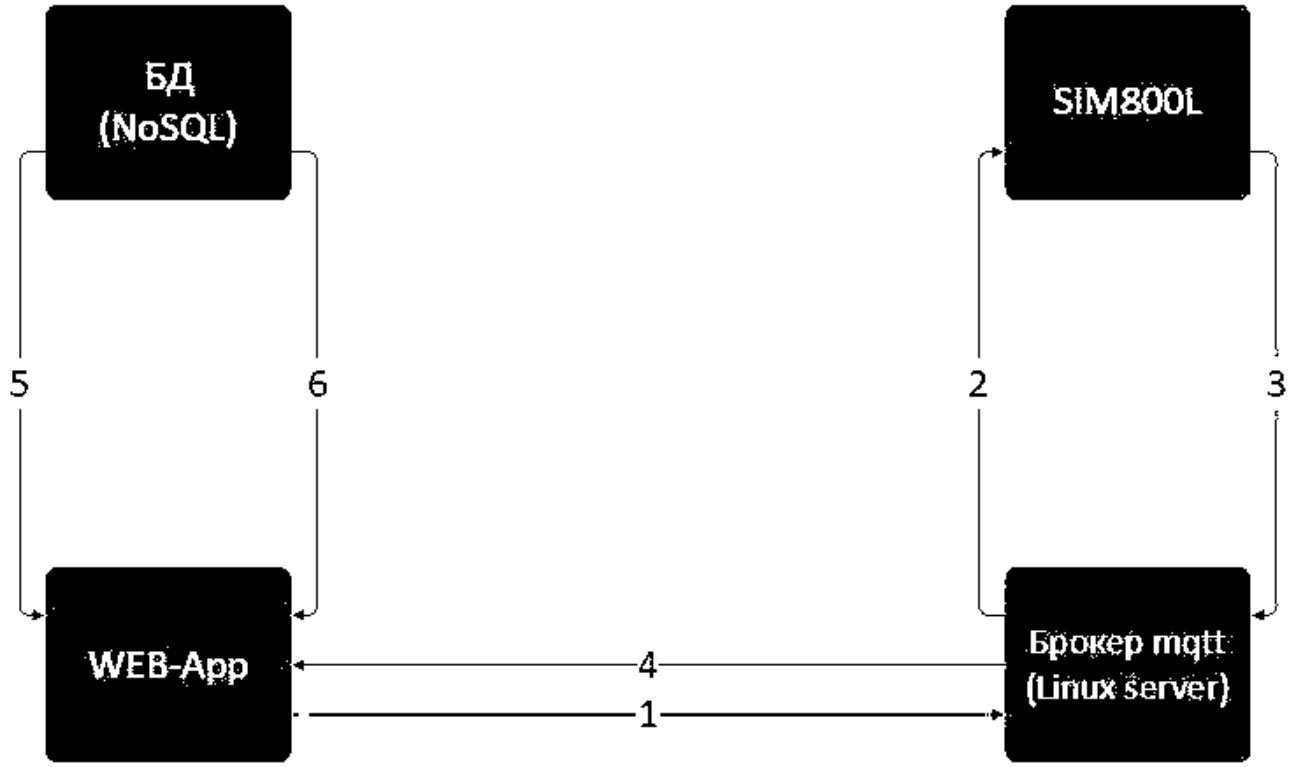


Рисунок 3.3 Схема передачі даних в системі

Опис схеми:

1)Користувач змінює параметри в додатку. Додаток надає брокеру параметри.

2)Запускається модуль та виконуються команди

3) Дані з модуля відправляються на брокер.

4)Брокер відає повідомлення на веб-сервер.

5)Веб-сервер записує дані в базу даних.

6)Дані з бази даних виводяться на Google maps

Дані передаються в JSON-форматі.

Рисунок 3.4 Вихід і інтернет

Модем завантажується та виконує AT команди реєструючись на мобільній базовій станції тобто сім отримує IP та доступ в інтернет.

Рисунок 3.4 Передача даних

3.2 Взаємодія фреймворків Flask та Dash.

Dash разом з Flask з точки зору структури проекту майже не відрізняється від базової структури додатка Flask.

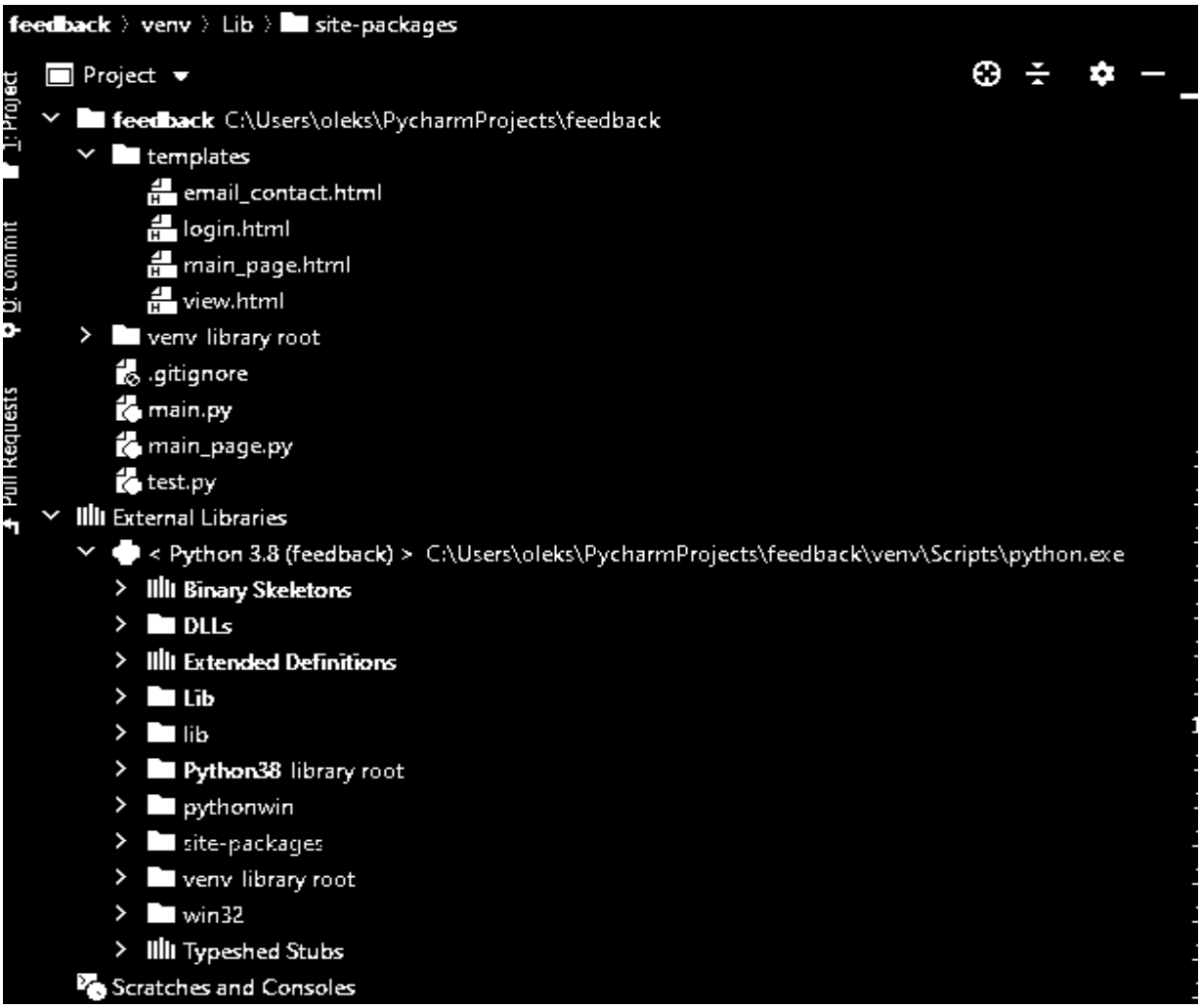


Рисунок 3.4 Структура проекта

Адмін панель містить форму для логіну та паролю. Логін і пароль задається в параметрах веб-додатку. У веб-форму яка написана на html увести дані у функцію передаються дані та виконується перевірка даних.

```python
@app.route('/login', methods=['GET','POST'])
def login():
    error = None
    if request.method == 'POST':
        if request.form['username'] != app.config['USERNAME']:
            error = 'Invalid username'
        elif request.form['password'] != app.config['PASSWORD']:
            error = 'Invalid password'
        else:
            session['Logged_in'] = True
            flash('You were logged in')
            return redirect(url_for('show_post'))
    return render_template('login.html', error=error)


@app.route('/logout')
def logout():
    session.clear()
    flash('You were logged out')
    return redirect(url_for('show_post'))
```

Рисунок 3.5 Код для авторизації

```
<main>
    <div class="form_admin_page">
        <form action={{url_for('login')}} method = post>
            <div class = "in_text">
                <label>Логін:</label>
                <input type = "text" name="username" required>
            </div>
            <div class = "in_text">
                <label>Пароль:</label>
                <input type = "password" name = "password" required >
            </div>
                <input type="submit" value="Підтвердити" class="button_send">
        </form>
    </div>


            <!--{% if error %}<p class = new3><strong>Error:</strong>{{ error}}{% endif %}-->
</main>
```

Рисунок 3.6 Розмітка для адмін панелі

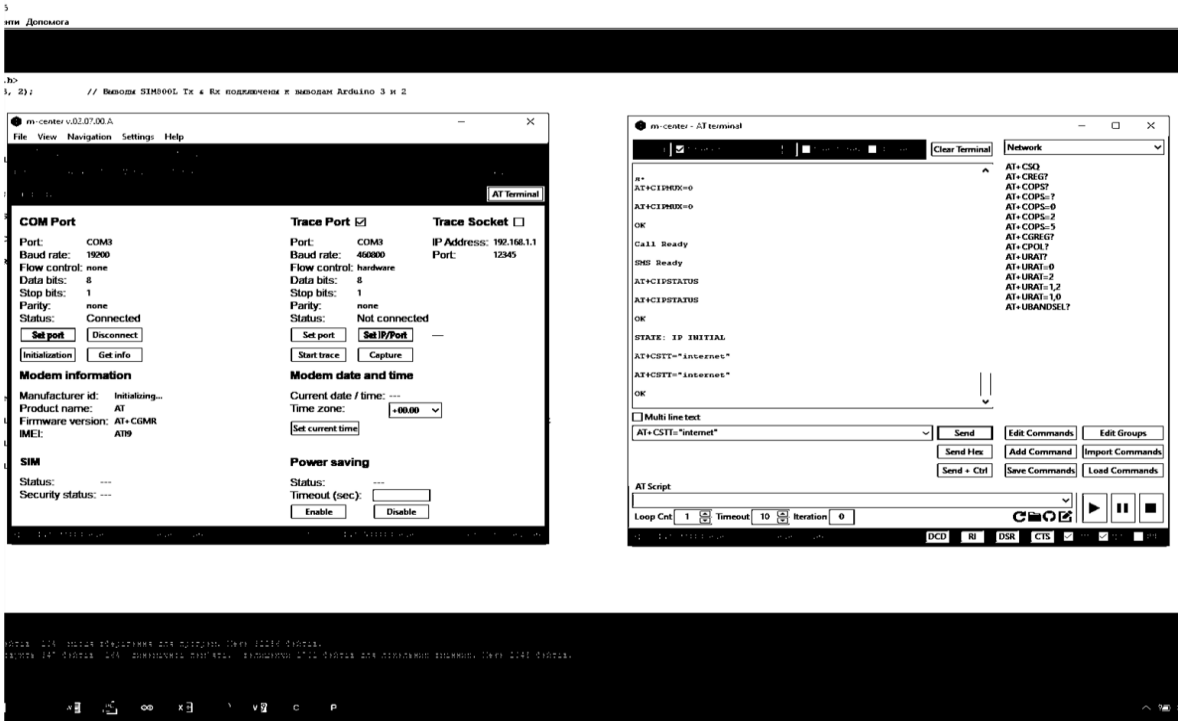


Рисунок 3.7 Відпрацювання AT команд

```python
@app.route('/feedback', methods=['POST', 'GET'])
def feedback_page():
    sender_email = os.environ.get('email_gmail')
    receiver_email = 'olleksii112@gmail.com'
    password = os.environ.get('password_gmail')
    if request.method == 'POST':
        with smtplib.SMTP('smtp.gmail.com',587) as server:

            server.ehlo()
            server.starttls()
            server.ehlo()

            server.login(sender_email,password)
            subject = request.form['subject_feedback']
            body = request.form['content']
            email_feedback = request.form['email_feedback']
            name_feedback = request.form['Name']
            msg = f'Subject: {subject}\n\n{email_feedback}\n\n{name_feedback}\n\n{body}'.encode('utf-8')

            server.sendmail(sender_email,receiver_email,msg)

    return render_template('email_contact.html')
```

Рисунок 3.10 Код для обробки feedback.

На рисунку 3.2 «Код для обробки feedback» вказаний код для обробки даних та надсилання їх на пошту gmail через бібліотеку smtplib. Для надсилання повідомлення на пошту потрібно вказати конфіденційні параметри такі як пароль

та логін до пошти. Пароль та логін я заховав використавши віртуальне середовище для цього.

Даний сервер використовує порт 587.

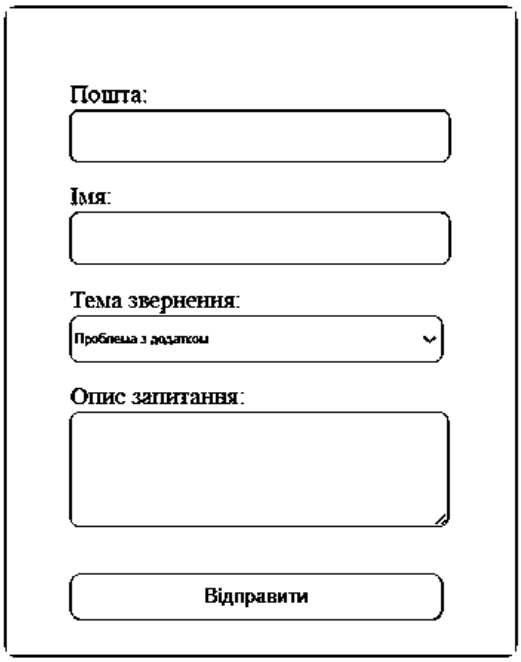


Рисунок 3.11 Відображення сторінки feedback

```
C:\Users\oleks\AppData\Local\Programs\Python\Python38\python.exe C:/Users/oleks/PycharmProjects/WEB_app/Program.py
 * Serving Flask app "Program" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 870-595-234
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [27/Dec/2022 04:03:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Dec/2022 04:03:25] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 - - [27/Dec/2022 04:03:26] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [27/Dec/2022 04:03:30] "GET /logout HTTP/1.1" 302 -
127.0.0.1 - - [27/Dec/2022 04:03:30] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Dec/2022 04:03:34] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [27/Dec/2022 04:03:42] "GET /add_2 HTTP/1.1" 200 -
127.0.0.1 - - [27/Dec/2022 04:03:43] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [27/Dec/2022 04:03:50] "POST /login HTTP/1.1" 302 -
127.0.0.1 - - [27/Dec/2022 04:03:50] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Dec/2022 04:03:52] "POST /delete HTTP/1.1" 500 -
Traceback (most recent call last):
```

Рисунок 3.12 Відповідь сервера

3.3 Опис бази даних.

У даному проекті я використовував MongoDB. Моя база даних складається з колекцій отриманих координат по певних обєктах.

Наприклад в мікроконтролер надсилає дані з модема в мікроконтролері ардуіно зашитий скетч в яких прописані відповідні команди для збору даних. Мікроконтролер групує інформацію та у вказаному форматі по певному topic через MQTT брокер передає дані через мобільну мережу які потім записуються в базу даних.

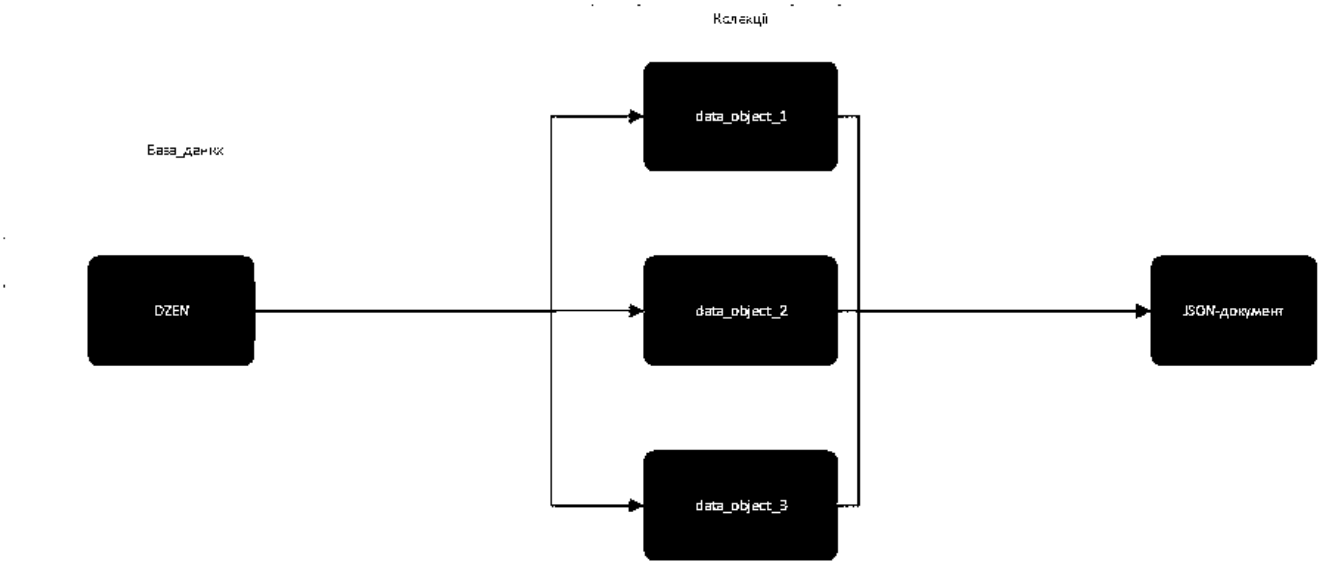


Рисунок 3.13 Діагарма структури бази даних

MongoDB server я успішно розвернув на комп'ютері та використовував локально даний сервер.

```
client = MongoClient('localhost', 27017)
db = client['DZEN']
db_collection = db['data_object']

broker_address = '192.168.112.92'
mqtt_client = mqtt.Client("P1")
mqtt_client.connect(broker_address)

data = pd.DataFrame(db_collection.find())
```

Рисунок 3.14 Підключення до сервера бази даних

Дані з модема приходять на мікроконтролер який групує їх в JSON формат та передає по певному topic на брокер сервер. Кожний topic по якому приходять

дані записується в базу даних. В даному проекті використовувалась така бібліотека як paho-mqtt-client.

Для брокера MQTT я локально розвернув віртуальну машину з операційною системою Linux та настроїв мережевий міст для того, щоб інші пристрої могли взаємодіяти з системою. Щоб під'єднатися до брокера потрібно прописати broker_address = 'IP'.

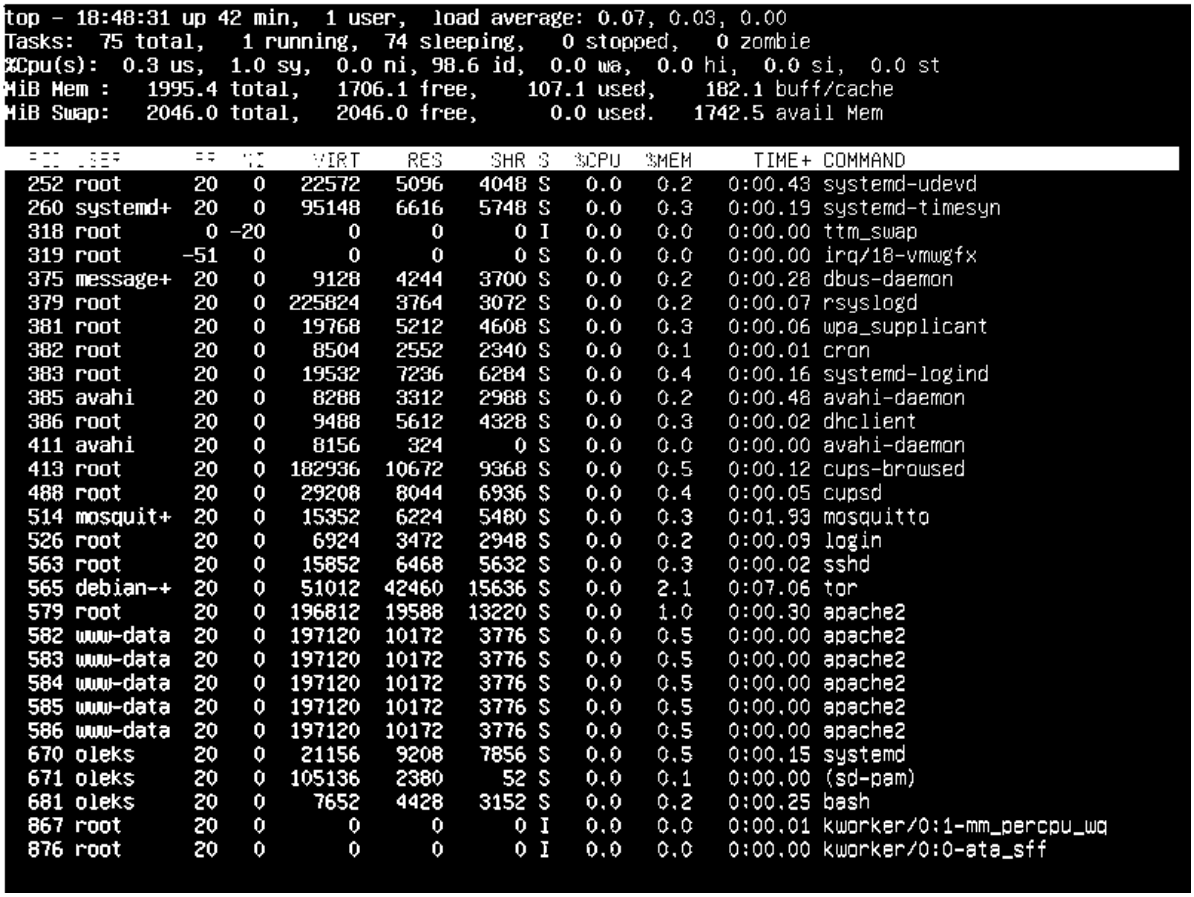


Рисунок 3.15 Процеси Linux

Виконавши команду top в операційній системі Linux будуть надані для перегляду. По індексу процесу під номером 514 можна переглянути що брокер працює.

3.3 Структура апаратної частини.

Рисунок 3.16 Процесор SIM800L

Через Arduino UNO включений через 2 та 3 цифрові піни як RX та TX. Також живлення 5V та GND.

В Arduino UNO зашитий скетч для відпрацювання АТ-команд. Виконавши команди формується JSON файл який передається по MQTT на брокер-сервер.

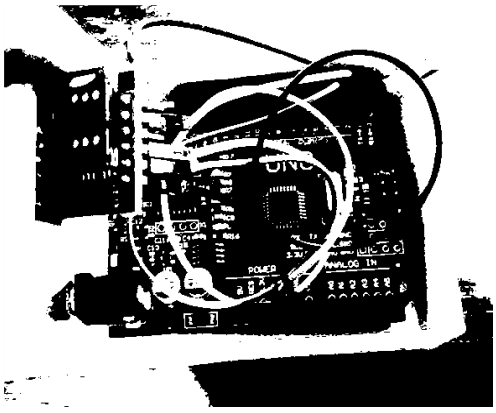


Рисунок 3.17 Апаратна частина

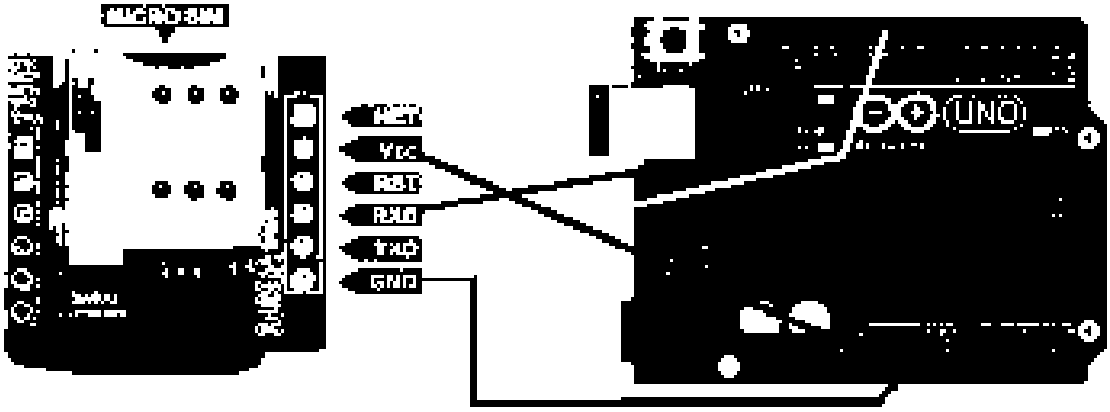


Рисунок 3.18 Схема підключення

Рисунок 3.18 Антенна

Довжина антени 48мм. Посилення 3dBi. Підходить для GSM модуля SIM800L. Призначена для пристроїв GSM (діапазон 900/1800 МГц)

ВИСНОВКИ

Було проаналізовано та рзроблено систему відстеження координат об’єктів на основі модулів GSM, GPRS з використанням Web-технологій потребує удосконалення корпусу для запобігання корозії та іншиш впливів на корпус. Дану систему надалі можна допрацювати зручніше для користувача. Також потрібно враховувати заряд АКБ для управління даною системою, інакше система моніторингу не зможе отримувати координати.

Для завантаження плати SIM800L потрібний час для підключення щоб виконати послідовність команд. Відхилення координат в декілька метрів можна пояснити наявністю неправильних отриманих даних з датчиків, але мало значущих для проведення моніторингу факторів.

.

ПЕРЕЛІК ПОСИЛАНЬ

1. Virtual environments Python [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/venv.html.
2. Flask documentation [Електронний ресурс] – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/2.0.x/foreword/#what-does-micro-mean.
3. Perry Lea. Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security / Perry Lea., 2018. – 524 с.
4. Michael McRoberts. Beginning Arduino / Michael McRoberts., 2010. – 472 с
5. MongoDB 4.4 Manual [Електронний ресурс] – Режим доступу до ресурсу: https://docs.mongodb.com/manual/.
6. Paho MQTT Python client library [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/paho-mqtt/#publishing.
7. Internet of Things with Python [Електронний ресурс] – Режим доступу до ресурсу: https://svitla.com/blog/internet-of-things-with-python.
8. Miguel Grinberg. Flask Web Development: Developing Web Applications with Python / Miguel Grinberg.. – 316 с.
9. Interface a SIM800L GSM GPRS module with Arduino [Електронний ресурс] – Режим доступу до ресурсу: https://www.electrovigyan.com/arduino/sim800l-module/.
10. В.Д. Муністер. Комп'ютерні мережі. IoT та міжмашинна взаємодія / В.Д. Муністер.
11. Plotly Python Open Source Graphing Library [Електронний ресурс] – Режим доступу до ресурсу: https://plotly.com/python/.
12. Jeremy Blum. Exploring Arduino: Tools and Techniques for Engineering Wizardry / Jeremy Blum., 2019. – 512 с.
13. SIM800+Series_TCPIP_Application+Note_V1.01 [Електронний ресурс] – Режим доступу до ресурсу: https://pdf1.alldatasheet.com/datasheet-pdf/view/1179001/ETC2/SIM800.html.

14. AT commands exemples [Електронний ресурс] – Режим доступу до ресурсу: https://content.u-blox.com/sites/default/files/AT-CommandsExamples_AppNote_%28UBX-13001820%29.pdf.
15. MicroPython libraries [Електронний ресурс] – Режим доступу до ресурсу: https://docs.openmv.io/library/index.html.
16. SIM800H (GSM/GPRS Module) Hardware Design [Електронний ресурс] – Режим доступу до ресурсу: https://datasheet-pdf.com/PDF/SIM800L-Datasheet-SIMCom-989664.
17. SIM808 high performance GSM/GPRS engine [Електронний ресурс] – Режим доступу до ресурсу: https://datasheet-pdf.com/PDF/SIM808-Datasheet-SIMCom-840127.
18. SIM800 Datasheet [Електронний ресурс] – Режим доступу до ресурсу: https://www.alldatasheet.com/datasheet-pdf/pdf/1179001/ETC2/SIM800.html.

Додаток А

Основний скетч Arduino IDE

```
// Select your modem:
#define TINY_GSM_MODEM_SIM800

// Set serial for debug console (to the Serial Monitor, default speed 115200)
#define SerialMon Serial

// Set serial for AT commands (to the module)
// Use Hardware Serial on Mega, Leonardo, Micro
#ifndef __AVR_ATmega328P__
#define SerialAT Serial1

// or Software Serial on Uno, Nano
#else
#include <SoftwareSerial.h>
SoftwareSerial SerialAT(2, 3);  // RX, TX
#endif

// See all AT commands, if wanted
// #define DUMP_AT_COMMANDS

// Define the serial console for debug prints, if needed
#define TINY_GSM_DEBUG SerialMon

// Add a reception delay, if needed.
// This may be needed for a fast processor at a slow baud rate.
#define TINY_GSM_YIELD() { delay(2); }
```

```
// Your GPRS credentials, if any
const char apn[]      = "internet";
const char gprsUser[] = "";
const char gprsPass[] = "";

// MQTT details
const char* broker = "IP_linux";

#include <TinyGsmClient.h>
#include <PubSubClient.h>

#define PUB_DELAY (5 * 1000) /* 5 seconds */

#ifdef DUMP_AT_COMMANDS
#include <StreamDebugger.h>
StreamDebugger debugger(SerialAT, SerialMon);
TinyGsm       modem(debugger);
#else
TinyGsm       modem(SerialAT);
#endif
TinyGsmClient client(modem);
PubSubClient  mqtt(client);

uint32_t lastReconnectAttempt = 0;

void mqttCallback(char* topic, byte* payload, unsigned int len) {
  SerialMon.println(topic);
  SerialMon.write(payload, len);
  SerialMon.println();
}
```

```
boolean mqttConnect() {
  SerialMon.print("Connecting to ");
  SerialMon.print(broker);

  // Connect to MQTT Broker
  boolean status = mqtt.connect("uno_sim800");

  if (status == false) {
    SerialMon.println(" fail");
    return false;
  }
  SerialMon.println(" success");

  mqtt.subscribe("base/relay/led1");
  return mqtt.connected();
}


void setup() {
  // Set console baud rate
  SerialMon.begin(115200);
  delay(10);

  SerialMon.println("Wait...");

  // Set GSM module baud rate
  SerialAT.begin(9600);
  delay(6000);
```

```
// Restart takes quite some time
// To skip it, call init() instead of restart()
SerialMon.println("Initializing modem...");
modem.restart();
// modem.init();

String modemInfo = modem.getModemInfo();
SerialMon.print("Modem Info: ");
SerialMon.println(modemInfo);

SerialMon.print("Waiting for network...");
if (!modem.waitForNetwork()) {
  SerialMon.println(" fail");
  delay(10000);
  return;
}
SerialMon.println(" success");

if (modem.isNetworkConnected()) {
  SerialMon.println("Network connected");
}

// GPRS connection parameters are usually set after network registration
SerialMon.print(F("Connecting to "));
SerialMon.print(apn);
if (!modem.gprsConnect(apn, gprsUser, gprsPass)) {
  SerialMon.println(" fail");
  delay(10000);
  return;
}
```

```
  SerialMon.println(" success");

  if (modem.isGprsConnected()) {
    SerialMon.println("GPRS connected");
  }

  // MQTT Broker setup
  mqtt.setServer(broker, 1883);
  mqtt.setCallback([] (char* topic, byte * payload, unsigned int len)  {
    SerialMon.write(payload, len);
    SerialMon.println();
  });
}


long last = 0;
void publishTemperature() {
  long now = millis();
  if (now - last > PUB_DELAY) {
    mqtt.publish("base/state/temperature", String(random(20, 30)).c_str());
    mqtt.publish("base/state/humidity", String(random(40, 90)).c_str());
    last = now;
  }
}

void loop() {
  // Make sure we're still registered on the network
  if (!modem.isNetworkConnected()) {
    SerialMon.println("Network disconnected");
    if (!modem.waitForNetwork(180000L, true)) {
```

```
    SerialMon.println(" fail");
    delay(10000);
    return;
  }
  if (modem.isNetworkConnected()) {
    SerialMon.println("Network re-connected");
  }

  // and make sure GPRS/EPS is still connected
  if (!modem.isGprsConnected()) {
    SerialMon.println("GPRS disconnected!");
    SerialMon.print(F("Connecting to "));
    SerialMon.print(apn);
    if (!modem.gprsConnect(apn, gprsUser, gprsPass)) {
      SerialMon.println(" fail");
      delay(10000);
      return;
    }
    if (modem.isGprsConnected()) {
      SerialMon.println("GPRS reconnected");
    }
  }
}

if (!mqtt.connected()) {
  SerialMon.println("=== MQTT NOT CONNECTED ===");
  // Reconnect every 10 seconds
  uint32_t t = millis();
  if (t - lastReconnectAttempt > 10000L) {
    lastReconnectAttempt = t;
```

```
        if (mqttConnect()) {
          lastReconnectAttempt = 0;
        }
      }
      delay(100);
      return;
    }

    mqtt.loop();
    publishTemperature();
  }
```

Скетч перевірки SIM800L.

```
#include <SIM800L.h>

#include <SoftwareSerial.h>

int t = 2;
int r =3;
//Create software serial object to communicate with SIM800L
SoftwareSerial mySerial(t, r); //SIM800L Tx & Rx is connected to Arduino #3 & #2

void setup()
{
  //Begin serial communication with Arduino and Arduino IDE (Serial Monitor)
  Serial.begin(9600);

  //Begin serial communication with Arduino and SIM800L
  mySerial.begin(9600);

  Serial.println("Initializing...");
```

```
  delay(1000);

  mySerial.println("AT"); //Once the handshake test is successful, it will back to OK
  updateSerial();
  mySerial.println("AT+CSQ"); //Signal quality test, value range is 0-31 , 31 is the best
  updateSerial();
  mySerial.println("AT+CCID"); //Read SIM information to confirm whether the SIM is
plugged
  updateSerial();
  mySerial.println("AT+CREG?"); //Check whether it has registered in the network
  updateSerial();
}

void loop()
{
  updateSerial();
}

void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
    mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port
  }
  while(mySerial.available())
  {
    Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port
  }
}
```

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення автоматизованих систем

МАГІСТЕРСЬКА РОБОТА

на тему:

«Розробка системи відстеження координат об’єктів отриманих за допомогою модулів GSM, GPRS з використанням Web-технологій»

Студент: Олексієнко О.С.

Керівник: Тушич А.М.

м. Київ 2022 рік

Мета роботи

Мета роботи - розробка веб-додатку для системи відстеженя координат за допомогою технологій GSM, GPRS та мови програмування Python.
Фреймворки Python використовуються для створення адмін панелі для авторизації користувача та відображення статистики та даних координат як на карті, так і в табличній формі.

Об’єкт дослідження - реалізація веб-додатка для системи відстежування координат об'єктів.

Предмет дослідження - предметом дослідження є система відстежування та відображенням координат у веб-додатку за допомогою мови Python.

Досліджувані питання магістерської роботи:

1. Аналіз системи відстеження координат за допомогою технологій GSM, GPRS;
2. Дослідження способів та засобів які застосовуються для реалізації системи відстеження координат за допомогою технологій GSM, GPRS.
3. Практична реалізація системи відстеження координат.

Аналіз системи

Розгортаючи систему відстеження координат об’єктів взаємодіють різні технології. Наприклад як GSM, GPRS, GPS, різні мови програмування та апаратна частина як трекери для відстежування до Ublox NEO-6M на базі Arduino.

Рисунок 1. Ublox NEO-6M

SIM800L

Модуль GSM, GPRS на чіпі SIM800L – це мініатюрний GSM-модем.

За його допомогою можна, відправляти SMS повідомлення, здійснювати чи приймати телефонні дзвінки, підключатися до Інтернету через GPRS, TCP/IP та багато іншого. А також модуль підтримує чотири діапазонну мережу GSM/GPRS.

Рисунок 1. SIM800L

Передача даних

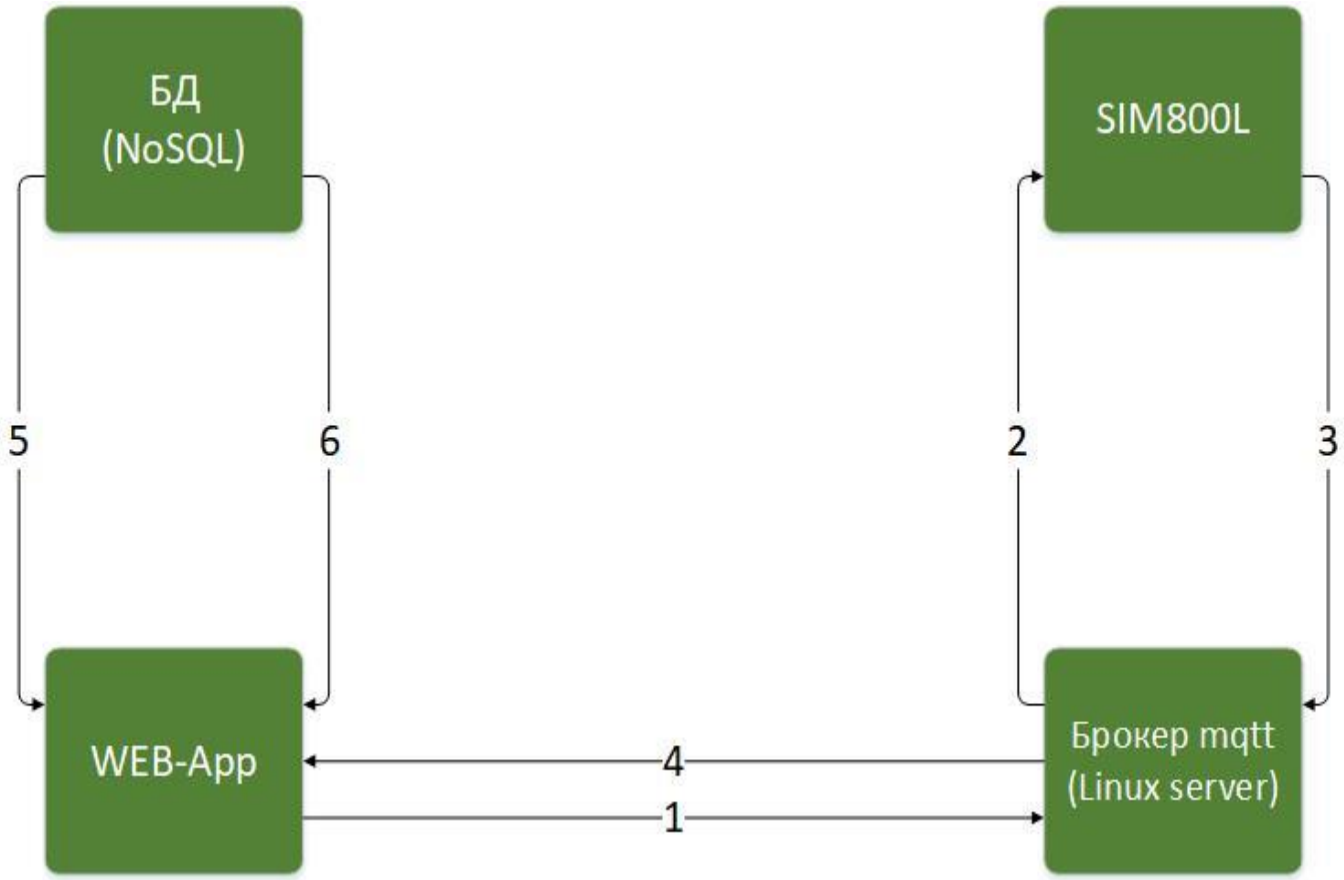


Рисунок 1. Схема передачі даних

1)Користувач змінює параметри в додатку. Додаток надає брокеру параметри.

2)Запускається модуль та виконуються команди

3) Дані з модуля відправляються на брокер.

4)Брокер відає повідомлення на веб-сервер.

5)Веб-сервер записує дані в базу даних.

6)Дані з бази даних виводяться на Google maps

Дані передаються в JSON-форматі.

Реалізація апаратної частини

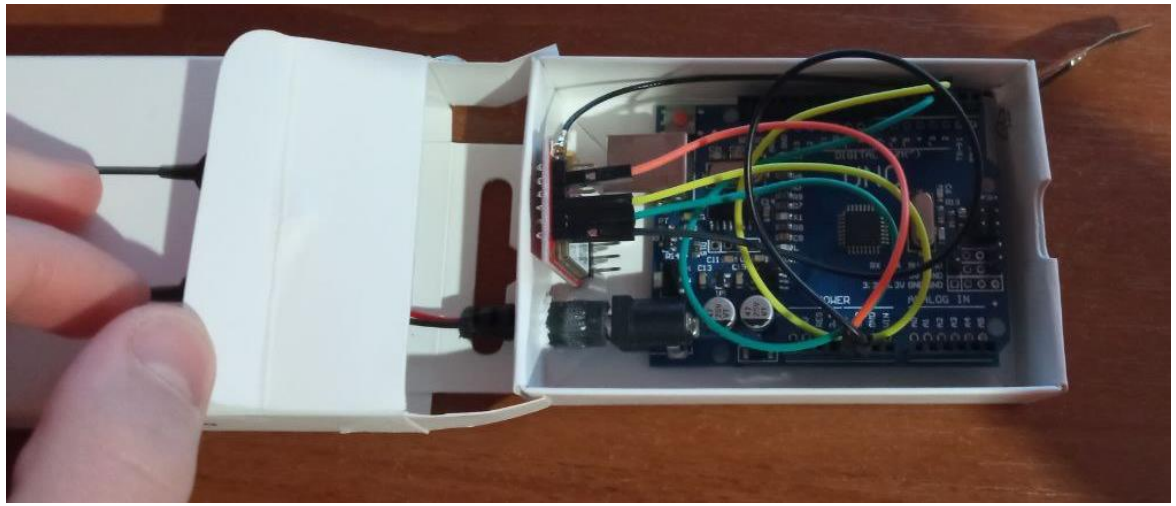


Рисунок 1. Апаратна частин

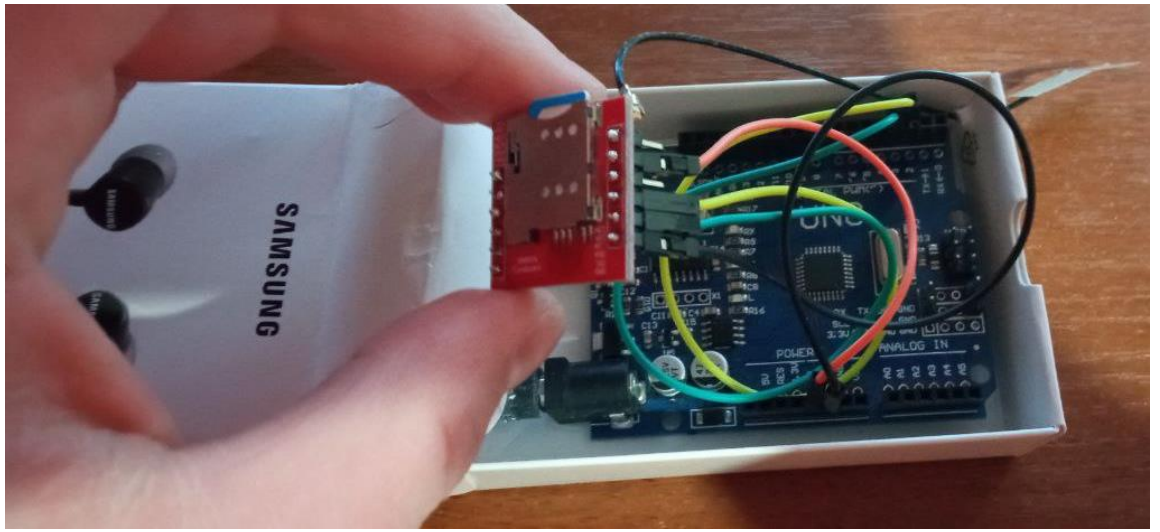
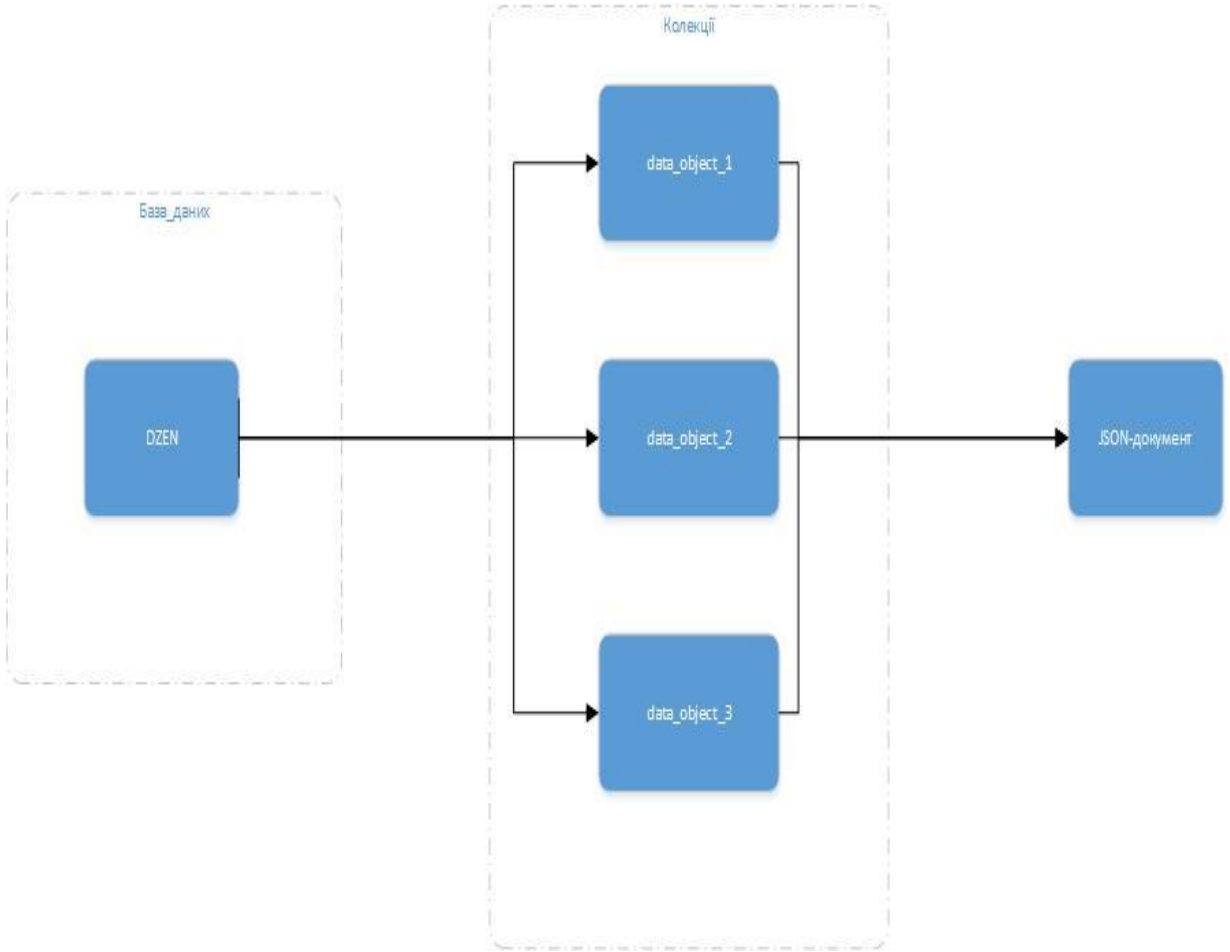


Рисунок 2. Плата SIM800L

Програмна реалізація елементів системи

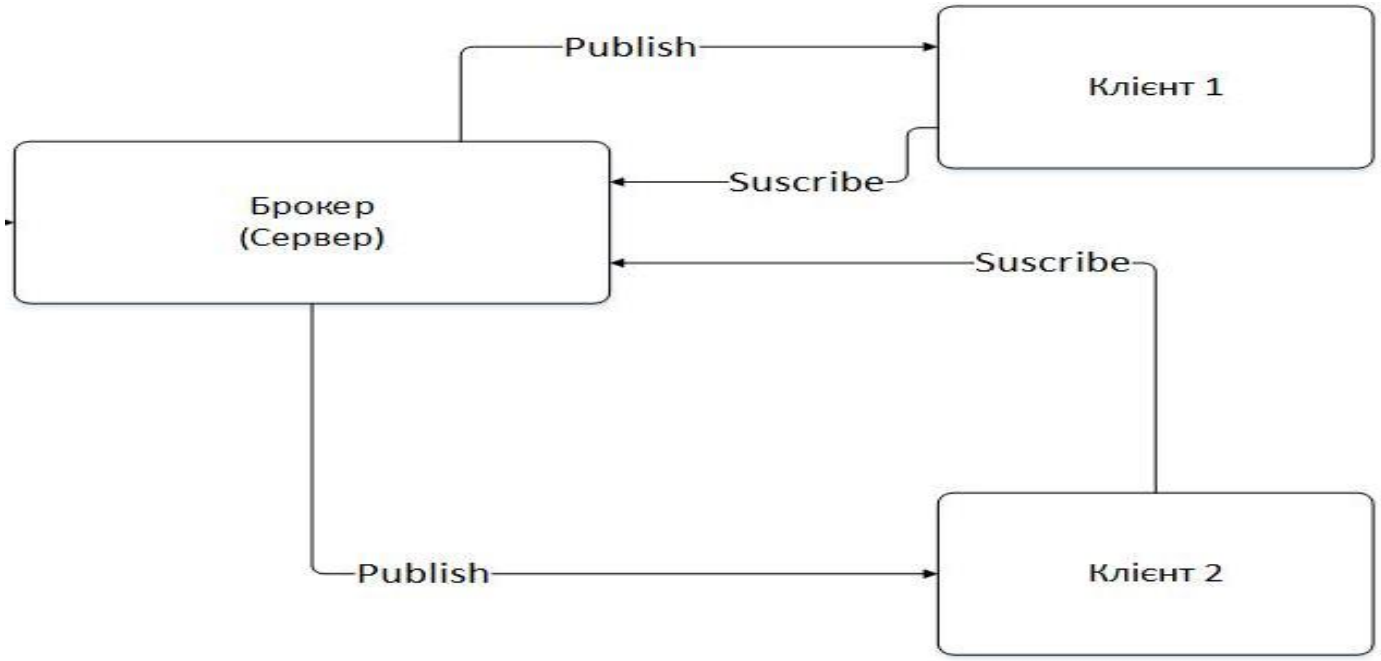


В даному проекті використовувалась база даних MongoDB.

Перегляд даних здійснюється через MongoDB Compass.

Рисунок 1. Діаграма структури бази даних

Програмна реалізація елементів системи



Eclipse Mosquitto - це посередник повідомлень із відкритим кодом (ліцензований EPL / EDL), який реалізує протокол MQTT. Брокер був встановлення на сервер Linux.

Рисунок 1. Опис передачі через mqtt-брокер

Координати

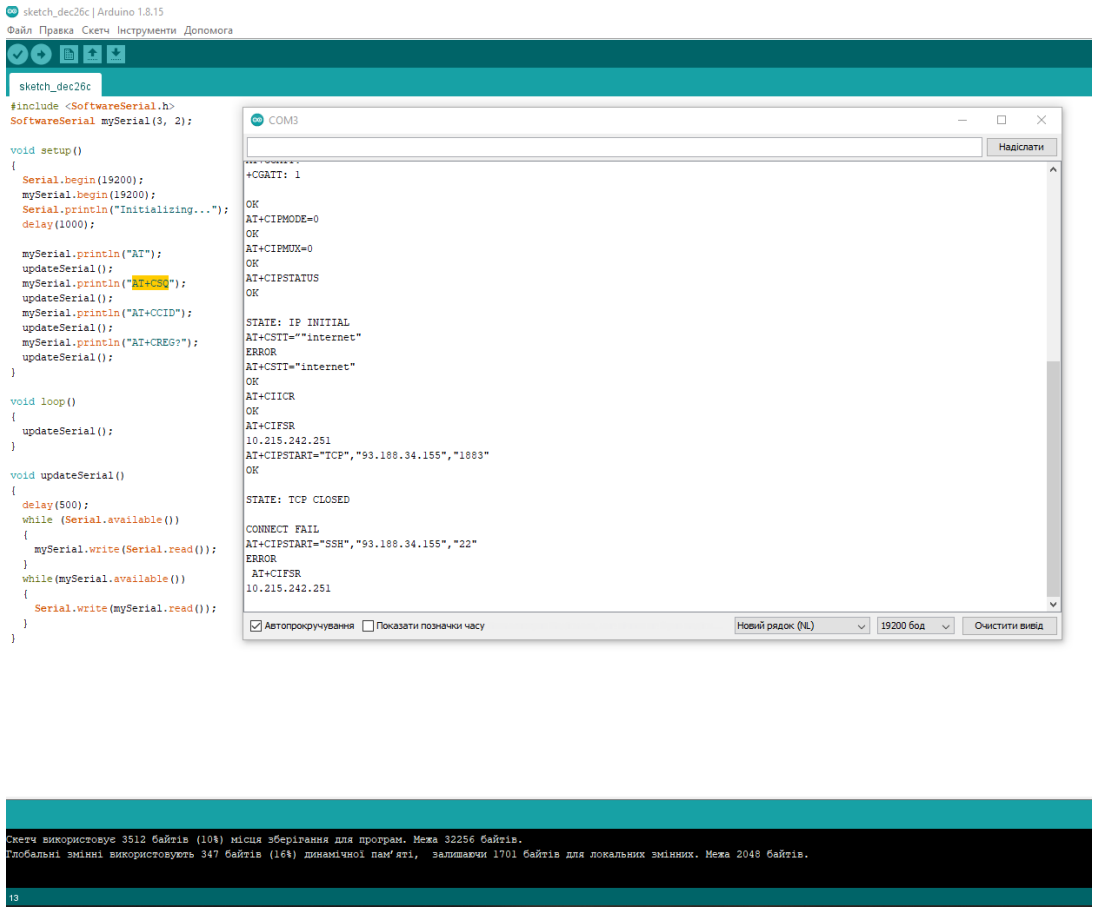


Рисунок 1. Монітор порта

Команда «AT+CLBS» має отримати розташування базової станції. Він використовується в поєднанні з «AT+CLBSCFG», що є конфігурацією для CLBS. Для визначення місцезнаходження SIM800.

GPS-відстеження та GSM-відстеження здійснюються приймачем, який збирає дані щонайменше з 4 супутників, щоб визначити точне положення. Пристрої відстеження GPS GSM виконують це завдання, посилаючись на інформацію з вишки стільникового зв'язку, яка є найближчою до пристрою відстеження GSM/GPS.

Висновки

Було проаналізовано та рзроблено систему відстеження координат об’єктів на основі модулів GSM, GPRS з використанням Web-технологій потребує удосконалення корпусу для запобігання корозії та іншиш впливів на корпус. Дану систему надалі можна допрацювати зручніше для користувача. Також потрібно враховувати заряд АКБ для управління даною системою, інакше система моніторингу не зможе отримувати координати.

Для завантаження плати SIM800L потрібний час для підключення щоб виконати послідовність команд. Відхилення координат в декілька метрів можна пояснити наявністю неправильних отриманих даних з датчиків, але мало значущих для проведення моніторингу факторів.

Апробація результатів

Основні положення і результати роботи доповідались і обговорювались на двох науково-практичних конференціях:

Дякую за увагу!