КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет радіофізики, електроніки та комп'ютерних систем
Кафедра комп'ютерної інженерії

АЛГОРИТМ МІНІМІЗАЦІЇ ТРАФІКУ В IoT СИСТЕМАХ

Дипломна робота магістра
студента 2 року навчання
спеціальність: 123 «Комп'ютерна інженерія»
Маренича Андрія

Науковий керівник
Юрчик Юрій Костянтинович,
асистент кафедри комп'ютерної інженерії

Рецензент
Лебедев Денис Юрійович
к.т.н., доцент кафедри конструювання
електронно-обчислювальної апаратури, факультет електроніки,
Київський політехнічний інститут імені Ігоря Сікорського

До захисту допускаю: _____ Завідувач кафедрою
Юрій БОЙКО

Ухвалено на засіданні кафедри "____" _____ 2022 р., протокол № ___

Київ - 2022

РЕФЕРАТ

Обсяг роботи за об’ємом складає 43 сторінки, містить 9 рисунків, 4 додатки, використано 10 інформаційних джерел.

ІНТЕРНЕТ РЕЧЕЙ, АНОМАЛІЯ, ПРОТОКОЛИ ВЗАЄМОДІЇ, МЕТОДИ ДЕТЕКТУВАННЯ АНОМАЛІЙ, АЛГОРИТМИ ДЕТЕКТУВАННЯ АНОМАЛІЙ, АЛГОРИТМ МІНІМІЗАЦІЇ ТРАФІКУ, MQTT, АЛГОРИТМ СЕРЕДНЬОКВАДРАТИЧНИХ ВІДХИЛЕНЬ, СЕРЕДНЬОКВАДРАТИЧНЕ ВІДХИЛЕННЯ.

Актуальність роботи полягає в тому, що кількість IoT пристроїв постійно зростає. Внаслідок цього збільшується об’єм мережевого трафіку. Пропускна спроможність каналів зв’язку зростає меншими темпами. Для вирішення цієї проблеми актуальним є питання зменшення обсягу інформації, що передається між елементами IoT.

Метою дослідження є розробка алгоритму мінімізації трафіку на основі детектування аномалій в мережах IoT, що взаємодіють з клієнтом через протокол MQTT.

Методи розроблення : алгоритм середньоквадратичних відхилень, метод ручного створення програмного продукту за допомогою середовища IDLE Python. Інструменти розроблення: безкоштовне, вільно поширюване інтегроване середовище розробки IDLE Python, мова програмування Python, клієнт MQTT-fx.

Результати роботи : проведене порівняння необхідних алгоритмів детектування аномалій, на основі їх порівняння обрано відповідний алгоритм, що відповідає вимогам поставленої задачі, розроблено алгоритм мінімізації трафіку, що працює на основі детектування аномалій, побудовано модель взаємодії пристроїв з клієнтом по протоколу MQTT, проаналізовано позитивні

та негативні сторони IoT протоколів. Отримані результати показали придатність алгоритму для виконання задач мінімізації.

Розроблений алгоритм мінімізації трафіку на основі детектування аномалій може використовуватись в системах з великою кількість пристроїв, що мають обмежені ресурси і взаємодіють між собою по протоколу MQTT.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

IoT - Internet of things

AI - Artificial intelligence

ML - Machine learning

CoAP - Constrained Application Protocol

MQTT - Message Queue Telemetry Transport

XMPP - Extensible Messaging Presence Protocol

AMQP - Advanced Message Queuing Protocol

IIoT - Industial Internet of things

UDP - User Datagram Protocol

REST - Representational State Transfer

QoS - Quality of service

URI - Uniform Resource Identifier

HTTP - HyperText Transfer Protocol

XEP - XMPP Extension Protocols

ВСТУП

IoT розвивається швидкими темпами і його системи вже широко використовуються у всіх сферах життя. Можливість підключення M2M машина-до-машини, дозволяє створювати системи без участі людини, і спонукає до розгортання систем з великою кількістю пристроїв. Використання протоколів IoT надають можливість без зайвих турбот інтегрувати ці системи, так як кожний з них розроблений для виконання певної специфічної задачі. Відповідно до теперішніх тенденцій, IoT – це глобальна інфраструктура, що включає в себе велику кількість пристроїв, взаємодія між якими заснована на відповідних протоколах. Через те, що кількість пристроїв невпинно зростає, а пропускна здатність каналів зв’язку зростає меншими темпами, виникає проблема з швидкістю передачі даних та їх обробкою і для її вирішення було розроблено відповідний алгоритм мінімазації на основі детектування аномалій, для системи, де пристрої взаємодіють по протоколу MQTT.

РОЗДІЛ 1 . ТЕХНОЛОГІЧНІ ЗАСАДИ ІНТЕРНЕТУ РЕЧЕЙ

1.1 Сучасний стан розвитку IoT

Створення «Інтернету речей», розробки нових пристроїв передачі інформації та покращення їх можливостей для обробки великих даних посприяли розвиненню автоматизації різних сфер життя. Для прикладу, розвиток таких систем як «розумний будинок», надало можливість кожному автоматизувати власну житлову систему, а постійне поліпшення пристроїв Інтернету речей забезпечило створення пристроїв з вбудованим програмним забезпеченням, завдяки чому створення IoT систем стало легким як ніколи.

Яскравим прикладом технології «Інтернету речей» є його впровадження в таких галузях як електрифікація, сільське господарство, промисловість та ін. Завдяки датчикам котрі зв’язані між собою на даний момент можна контролювати великі системи без зайвих проблем, так як датчики температури надають можливість контролювати вологість і температуру в кімнаті чи на полях для покращення врожайності, можливості швидкої заміни ліній електропередач чи контролюванню температури в регіонах, інші типи датчиків надають можливості для контролювання тиску на різних етапах виробництва в галузі промисловості, датчики розгерметизації, руху тощо. Завдяки даним пристроям виникає позитивний крок для зменшення витрат на енергоресурси, додаткові витрати на виробництво і збільшення продуктивності різноманітних систем.

1.2 Протоколи взаємодії пристроїв

MQTT – протокол котрий створений для взаємодії пристроїв в системах (мережах) з низькою пропускною здатністю. Працює MQTT на основі TCP, що надає можливість додаткової надійності в обміні інформацією між пристроями IoT. Ключовою ціллю MQTT є моніторинг даних, що надходять від значної кількості пристроїв, що сприяє масштабуванню мереж до надвеликих розмірів, з можливістю віддаленого аналізу даних. [10]

Основною ідеєю протоколу MQTT є система, котра працює через брокера даних, так як надсилання і отримання даних відбувається саме до брокера і від нього. Брокер – це додаток чи програма котра отримує дані від пристроїв IoT і записує повідомлення до відповідної бази даних.

В системі, що працює на основі протоколу MQTT обмін данними відбувається між клієнтом та брокером.

Основними повідомленнями котрі використовуються для взаємодії з брокером-повідомлень це :

- connect – під’єднатись до брокера;
- publish – надіслати дані на topic в брокері;
- unsubscribe – відписатись від topic.
- disconnect – відключитись від брокера;
- subscribe – підписатись на топік брокера;

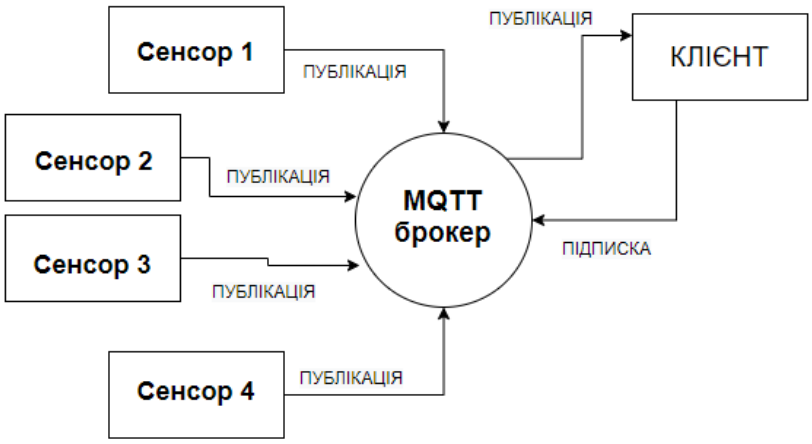


Рисунок 1 - Схема обміну повідомлень через MQTT брокер

Брокер налічує в собі таблицю котра складається з інформації про кожний пакет котрий був отриманий та спосіб ідентифікації даних пакетів по відповідним темам. (topic – (тема) визначений та з фіксованим значенням заголовок керуючого пакету). Система надсилання повідомлень від брокера

відбувається на такій основі, спочатку брокер отримує відповідне повідомлення після чого брокер надсилає повідомлення до кожного пристрою що має відповідну підписку на тему, тобто аби пристрій мав змогу отримати повідомлення йому наеобхідно мати підписку. Кожна тема створюється динамічно при події коли пристрій підписується або при отриманні пакету котрий має в заголовку визначену тему. Теми в загальному, це технологія котра дозволяє зручно організувати зв’язки в мережі.

Наприклад з’язки такого типу як:

- один до багатьох
- один до одного
- багато до багатьох

Може виникати і сиутація такого характеру, коли втрачено з’єднання з пристроєм котрий є підписником, то інформація про пакет видаляється з таблиці брокера, внаслідок чого необхідно необхідно розробити певний механізм для очікування на його підключення в залежності від заданих параметрів, ща надає можливість збільшити ефективність взаємодії пристроїв з брокером.

Для забезпечення максиальної стійкості механізму роботи протоколу MQTT можливим є використання трьох рівнів QoS, котрі забезпечують додаткову надійність при обміні повідомленнями в мережі, а саме :

- Відправити та забути - повідомлення надсилається лише один раз, і підтвердження доставки пакетів не гарантовано.
- Надсилається принаймі один раз - повідомлення надсилається принаймні один раз, в даному випадку необхідним є підтвердження отримання пакетів.

- Доставлено рівно один раз – в даному випадку використовується спосіб передачі інформації з використанням чотирьохстороннього рукостискання, що забезпечує гарантію доставки повідомлення рівно один раз.

Основні особливості протоколу MQTT :

- мала вага повідомлень;
- підтримка роботи системи в нестабільних мережах з малою пропускною здатністю;
- можливість використання чотирьох рівнів якості обслуговування QoS;
- підтримка масштабування мережі (збільшення пристроїв).

CoAP – протокол, що використовується в мережах котрі скаладаються з пристроїв які мають обмежені ресурси і споживають малу кількість електроенергії, через ці обмеження виникає проблема, рішення якої полягає в зменшенні ваги повідомлень для того, аби забезпечити достатню ефективність враховуючи відповідні критерії роботи даних елементів IoT. CoAP використовує для обміну повідомленнями між пристроями протокол HTTP, котрий полегшенує його сумісність з мережею Інтернет.

Мережа в котрій використовується протокол CoAP виглядає таким чином, в системі з використанням даного протоколу повинен існувати CoAP-пристрій, що отримує повідомлення від користувача, внаслідок чого сервер отримує запити від користувача після чого індивідуально надсилає відповідні запити котрі визначені для кожного з пристроїв мережі, і чекає поки пристрій надішле відповідь.

Основні особливості протоколу CoAP.

Структура CoAP була розроблена на основі архітектури REST. REST – тип архітектури, котрий надає можливість взаємодії між пристроями на основі таких методів.

- Метод GET – виконує зчитування даних відповідно до джерела котре визначене в URI-посиланні, як джерело даних можна привести до прикладу датчик, з якого ми будемо отримувати відповідну інформацію.
- Метод PUT – метод PUT використовується для зміни даних або їх виправлення внаслідок виникнення певних помилок за допомогою перезапису.
- Метод POST – метод POST служить для додавання нових ресурсів.
- Метод DELETE – використовується для видалення відповідних ресурсів та записів про них.
- Метод FETCH – надає змогу отримати інформацію про ресурс.
- Метод PATCH - відповідає за часткову зміну ресурсу.

Завдяки використанню даних методів протоколу HTTP, вирішуються проблеми з тим, щоби отримати інформацію або змінити певний ресурс лише частково, тобто змінити певну складову ресурсу. На відміну від MQTT, CoAP використовує UDP для обміну пакетами в мережі, це надає змогу зменшити вагу пакетів (розмір даних), що і є основною ціллю даного протоколу.

XMPP – протокол котрий працює на основі мови XML. Найстаріший протокол з наведеного списку, через, що був досить популярний в певний час, але був витіснений більш прогресивними рішеннями.[10]

XMPP працює на основі TCP, що надає додаткову надійність при обміні інформацією в мережі, як і в протоколі MQTT центральною технологією є використання брокера, що отримує та пересилає дані на основі функцій

публікації та підписки, на відміну від протоколів, котрі використовують функції запиту та відповіді як в CoAP. XMPP підтримує повідомлення з невеликою вагою та забезпечує обмін даними з низькою затримкою, а також підтримує розширення своєї функціональності завдяки специфікації XEP.

XMPP підтримує ту технологію, що і MQTT, завдяки чому їх принципи роботи достатньо схожі причому XMPP протокол підтримується в мережі Інтернет досить давно на відміну від новітньої технології MQTT. Але основним недоліком залишається використання XML-повідомлень, котрі створюють проблеми, які впливають на ефективність роботи системи через непотрібні теги, також необхідно враховувати додаткові витрати на вивчення XML-документа, що призводить до збільшення обчислювальної здатності пристроїв.

Основними перевагами протоколу є :

- Простота налаштування. Розгортка власного XMPP-сервера відбувається достатньо швидко.
- Підтримка в різних галузях мережі Інтернет.
- Налаштування функціональності, для підтримки можливості взаємодії різних мереж.
- Використання технології підписки/публікації.
- Відправлення повідомлень відбувається з низькою затримкою.

AMQP – протокол, котрий призначений для обміну інформацією між пристроями з низькою затримкою, а завдяки підтримці TCP, надає гарантію доставки повідомлень з підтвердженням отримання. Вагомою особивістю даного протоколу є його можливість розпаралелювати процеси за допомогою використання декількох серверів, що пришвидшує обробку повідомлень котрі мають помітку термінових.[10]

Центральною ідеєю технології є використання AMQP-брокера, завдяки чому пристрої IoT мережі мають змогу вільно обмінюватись повідомленнями через відповідні теми, з врахуванням гарантії доставки чи ні.

AMQP основними компонентами протоколу є :

- Повідомлення (message) – одиниця передачі даних, яка може містити в собі відповідні прикріплені заголовки.
- Черга (queue) – в черзі повідомлення зберігаються до певного моменту поки не буде необхідності в їх отриманні.
- Точка обміну (exchange) – даний компонент отримує повідомлення, після чого exchange в залежності від параметрів повідомлення, сортує повідомлення в одну або деяку кількість черг, але сама не зберігає їх.

  Всього визначено три типи точок обміну :

  - fanout – message надсилається в усі черги котрі закріплені за цією точкою обміну;
  - direct – message надсилається у чергу з визначеним ім’ям;
  - topic – message надсилається в чергу для якої співпадає заголовок котрий додається до ключа маршрутизації;

Завдяки використанню механізму черг, виникає можливість створювати системи котрі матимуть змогу отримувати необхідну інформацію без додаткових витрат на опитування компонентів мережі, тобто пристрій підписується на відповідну тему в черзі, завдяки чому він отримає повідомлення як тільки воно надійде на відповідну тему з черги.

1.3 Порівняняльна характеристика засобів взаємодії

Наведені протоколи IoT досить сильно відрізняються між собою, кожний з них призначений для вирішення специфічних задач, внаслідок чого потрібно

звертати увагу при їх обранні - на швидкість, контроль якості QoS, обробку даних, сумісність з іншими протоколами та споживання ресурсів.

Для того, аби визначити протокол котрий накраще підійде для виконання поставленої задачі потрібно врахувати швидкість обробки даних, можливість масштабування мережі, та підтримку мереж з можливими проблемами швикості передачі даних, XMPP в цьому випадку проявляє себе з гіршої сторони, так як використання XML-повідомлень потребує значого споживання ресурсів, що вже стає проблемою для задачі мінімізації в пристроях з обмеженими ресурсами. На відміну від нього, MQTT може справитись з задачею набагато краще завдяки своїй орієнтованості на нестабільні мережі з низькою пропускною здатністю, орієнтованістю на велику кількість пристроїв та підтримку QoS, та протоколу TCP, ща є вагомою перевагою над CoAP. AMQP завдяки орієнтованості на черги може працювати неймовірно швидко з мінімальними затримками, але може виникнути проблема з отриманням інформації коли затримка в мережі напряму почне впливати на черги.

Взявши до уваги різницю протоколів в підтримці сучасних топологій, можна зробити висновок, що CoAP, MQTT, XMPP, AMQP підтримують сумісність з різними протоколами та надають можливість легко налаштувати взаємодію IoT пристроїв в мережі, але врахувавши недоліки кожного з них найкращим протоколом для виконання поставленої задачі було обрано протокол MQTT, завдяки своїй швидкості, простоті налаштувань, орієнтованості на велику кількість підключених пристроїв, а також роботі в нестабільних мережах.

1.4 Проблема мінімізації трафіку в системах взаємодії пристроїв

На скільки не було б гарантовано надсилання повідомлень з датчика на сервер, безпека IoT-мереж, швидкість роботи протоколів, взаємодія з великою кількістю пристроїв, робота в нестабільних мережах – виникає проблема з

мінімізацією трафіку. Головною причиною виникнення даної проблеми є те, що кількість пристроїв постійно зростає досить швидкими темпами ніж мережеве обладнання через це необхідно розробити відповідний алгоритм, для вирішення даної задачі. З іншої точки зору, пристрої в залежності від типу, взаємодіють з різними середовищами і отримують інформацію різного типу з різними інтервалами, але зазвичай вона надходить в досить малі проміжки часу, внаслідок чого при нестабільній роботі мережі чи інших проблемах, особливо коли кількість пристроїв значна, надсилання даних може призвести до перевантаження мережі, тому і виникає необхідність в мінімізації даних, яку можна провести завдяки використанню алгоритмів мінімізації трафіку.

РОЗДІЛ 2 . АЛГОРИТМ МІНІМІЗАЦІЇ ТРАФІКУ В IoT СИСТЕМАХ

2.1 Поняття аномалії та її типи

Аномалія – це нетипове значення, котре має відхилення від певної норми для визначеного набору. На даний момент важливим є розробка алгоритмів та методів детектування аномалій та розробка на їх основві алгоритмів мінімізації трафіку. В загальному існує три типи аномальності даних серед яких визначені такі як: точкові, контекстуальні та колективні аномалії.

Точкові аномалії - даний тип аномалій представлений певним одиничним екземпляром котрий виділяється серед набору даних, як приклад аномальне значення температури (підвищення відносно норми) для кімнатного приміщення.

На рис. 4 показано як виглядають точкові аномалії. Серед неаномальних наборів значень m1 та m2 можна спостерігати певне аномальне точкове значення a, завдяки простоті самої аномалії її досить легко детектувати за допомогою вже існуючих алгоритмів детектування аномалій.

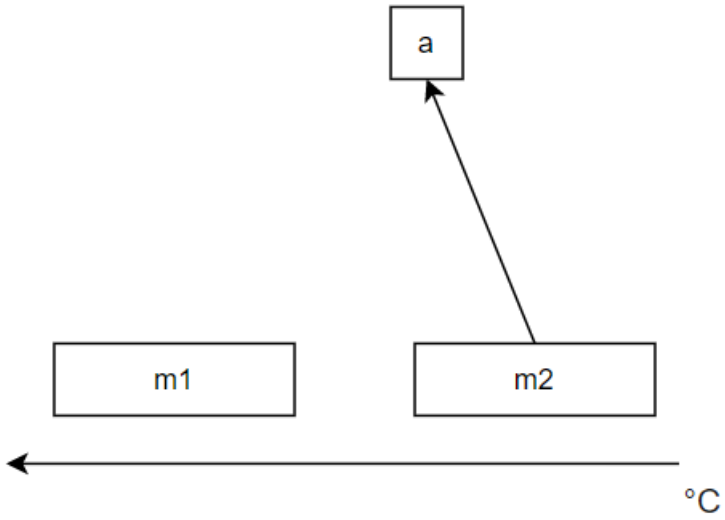


Рисунок 2 - Точкові аномалії

Контекстні аномалії - це тип аномалій, до яких відносяться певні екземпляри даних котрі вважаються аномальними лише за певних умов. Для визначення аномальності екземпляра необхідно врахувати контекстні та поведінкові параметри.

Контекстні параметри – це складові котрі необхідні для визначення під яку умову, що задана для набору значень підпадає той чи інший екземпляр даних, а поведінкові параметри визначаються неконтекстуальними характеристиками того ж екземпляра.

Аномальність екземпляра визначається на основі параметрів поведінки відповідно до заданих умов, тобто екземпляр може бути аномалією за даних умов, але за такими ж параметрами поведінки буде рахуватись неаномальним за іншої умови.

На рис. 5 можна побачити значення m1 – m3, котрі являються типовими, в той час як значення a1 є аномалією, хоч воно і має аналогічне значення з m1-m3. В цьому і полягає основна ідея контекстних аномалій з врахуванням контекстних та поведінкових параметрів.

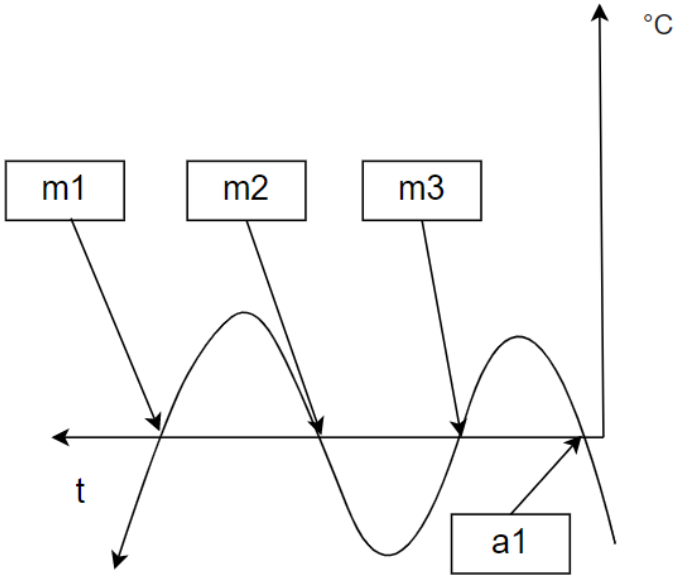


Рисунок 3 - Контекстні аномалії

Колективні аномалії – аномалії котрі складаються з декількох значень, і вважаються аномальними відносно всього набору даних, в таких випадках одиничне значення може не нести характер проблеми, але враховуючи інші значення виникає поняття колективної аномалії.

На рис. 6 проміжок m2 – m3 є колективною аномалією, серед набору m1.

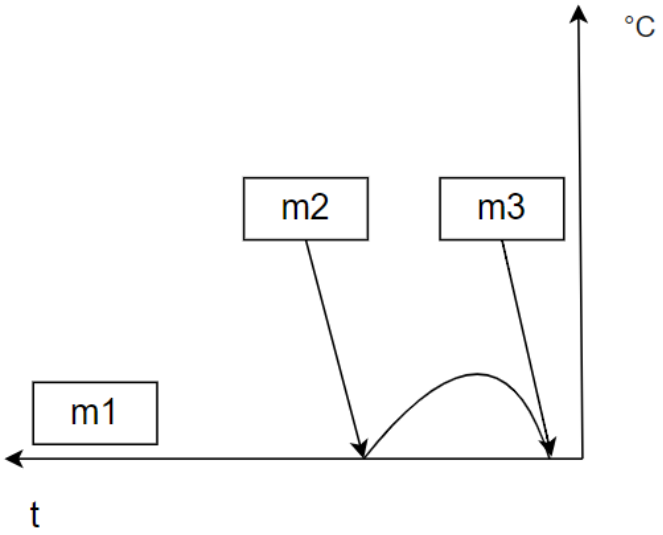


Рисунок 4 - Колективні аномалії

Важливою особливістю колективних аномалій є те, що вони не спостерігаються в будь-яких наборах даних як точкові чи контекстні аномалії, а лише в тих випадках де дані зв’язані між собою. Цікавим фактом є те, що точкові і контекстні аномалії можуть бути колективними.

Для розробки алгоритму мінімізації було розглянуто 3 види аномалій, завдяки чому відносно отриманих результатів роботи стало зрозумілим, що в реалізованій системі IoT виникали саме точкові аномалії, а отже можна говорити про спрощення реалізації алгоритму для використання в пристроях з обмеженими ресурсами.

2.2 Математичні алгоритми детектування аномалій

Середньоквадратичне відхилення - найбільш поширений показник розкиду значень випадкової величини щодо середнього значення більш відомого як математичне очікування.[5][7]

За визначенням середнє квадратичне відхилення є додатнім квадратним коренем із дисперсії. Дисперсія, характеризує розсіяння значень навколо центру розподілу, більшому значенню стандартного відхилення відповідає більший розкид значень. Перевагою стандартного відхилення в порівнянні з дисперсією полягає в тому, що розмірність стандартного відхилення рівна розмірності випадкової величини, а розмірність дисперсії це квадрат розмірності випадкової величини.

Середньоквадратичне відхилення та середньоарифметичне значення обчислюється за формулами:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n},$$

де $\bar{X}$ - середньоарифметичне значення;

x_i - і-ий елемент вибірки;

n - обсяг вибірки.

$$S = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n}},$$

де S – середньоквадратичне значення;

$\bar{X}$ – середньоарифметичне значення вибірки;

x_i – і-ий елемент вибірки;

n – обсяг вибірки.

В аналізі даних середньоквадратичне відхилення використовується як міра мінливості значень ознак, ступенів відхилення бажаних показників від спостережуваних, а також для визначення викидів і аномальних значень в даних за допомогою правила трьох сигм.

В аналізі даних середньоквадратичне відхилення використовується як міра ступенів відхилення очікуваних показників від спостережуваних, а також для визначення викидів і аномальних значень в даних завдяки правилу трьох сигм, воно означає, ймовірність того, що випадкова величина відхилиться від власного математичного очікування більш ніж на 3 середньоквадратичних відхилень, майже рівна нулю.

Тест по критерію x^2 (хі-квадрат) застосовується для перевірки того, що певна вибірка відповідає визначеному закону розподілу. Відносно правила трьох сигм для нормально розподіленої випадкової величини, яка відповідна виміру трафіка без аномалій, обчислюється верхня межа зміни її значення.

Аномалія фіксується у випадку перевищення цієї границі значенням x^2 спостережуваної величини.

Основний принцип даного методу полягає в критерії x^2 який виражений формулою :

$$x^2 = \sum_{i=1}^{N} \frac{(o_i - E_i)^2}{E_i},$$

де o_i - об'єкт тестування, o_i це значення o у вибірці i.

E_i - середнє значення вибірки i серед усіх об'єктів.

Об'єкт може бути ідентифікований як викид, якщо значення x^2 більше порогового значення.

Переваги

Висока точність знаходження аномалій у випадку нормально розподілених випадкових величини.

Недоліки

Можуть виникати проблеми в реалізації через обширність та складність формул визначення випадкової величини.

Необхідність задання найбільш повної вибірки вимірів типового трафіка.

Алгоритм порогового аналізу працює на основі задання діапазону змін значень спостережуваних параметрів. Якщо значення знаходиться за рамками цього діапазону, воно носитеме характер аномалії. Для того, щоб зменшити кількість помилкових спрацювань, метод можна покращити за допомогою лічильника, який буде накопичувати події «випадіння» спостережуваних параметрів з діапазону. При підвищенні лічильником визначеного значення, фіксується факт наявності аномалії.

Переваги.

Простота реалізації і налаштування, інтерпретації отриманих результатів, що свідчать про типовість/аномальність події в мережі.

Недоліки.

Відсутність адаптивних механізмів, для автоматичного вибору межі. Необхідність ретельного аналізу отриманих результатів внаслідок можливих помилкових спрацювань.

Серед даних алгоритмів детектування аномалій, було обрано алгоритм середньоквадратичних відхилень, так як легкість в обрахунках і адаптивному налаштуванні алгоритму для виконная різного роду задач, сприяють збільшенню сфер його використання, та зменшенню навантажень на пристрої з обмеженими ресурсами на відміну від інших алгоритмів.

2.3 Реалізація IoT системи

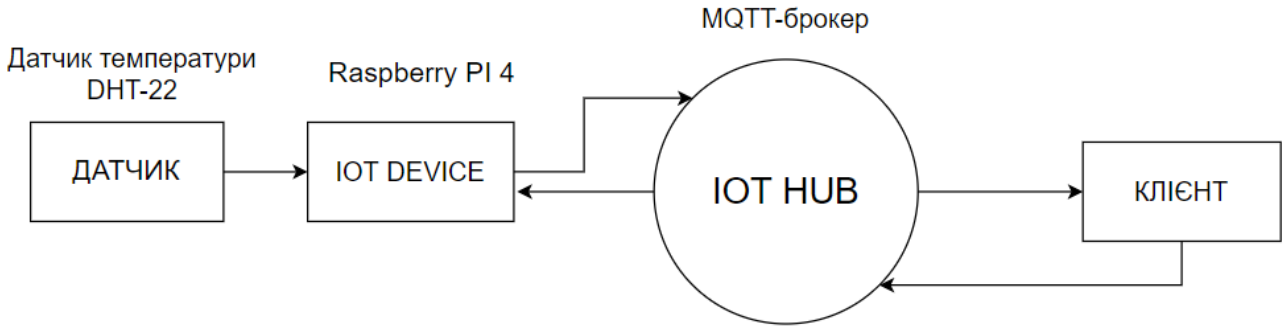


Рисунок 5 – Схема реалізованої IoT системи

Реалізована IoT система складається з таких компонентів :

- Датчик (модель DHT-22) – використовується дли вимірювання температури навколишнього середовища, джерело даних.

- IoT DEVICE (Raspberry PI 4)– плата до якої підключений датчик, саме на ній реалізовано алгоритм мінімізації трафіку на основі алгоритмів детектування аномалій .
- IoT HUB (OS Fedora server) – віртуальна машина, яка виконує функцію брокера.

Робота даної системи реалізована таким чином. Спочатку, дані надсилаються з датчина на IoT device, після чого починається робота програми, що заснована на алгоритмі середньоквадратичних відхилень, надалі дані надсилаються з IoT device на IoT HUB (який працює як MQTT-брокер) в мінімізованому вигляді, після чого надсилаються до клієнта. Обмін інформацією відбувається за допомогою протокола MQTT, датчик підписується на тему брокера після чого надсилає відповідні дані.

2.4 Моделювання алгоритму мінімізації на основі алгоритмів детектування аномалій

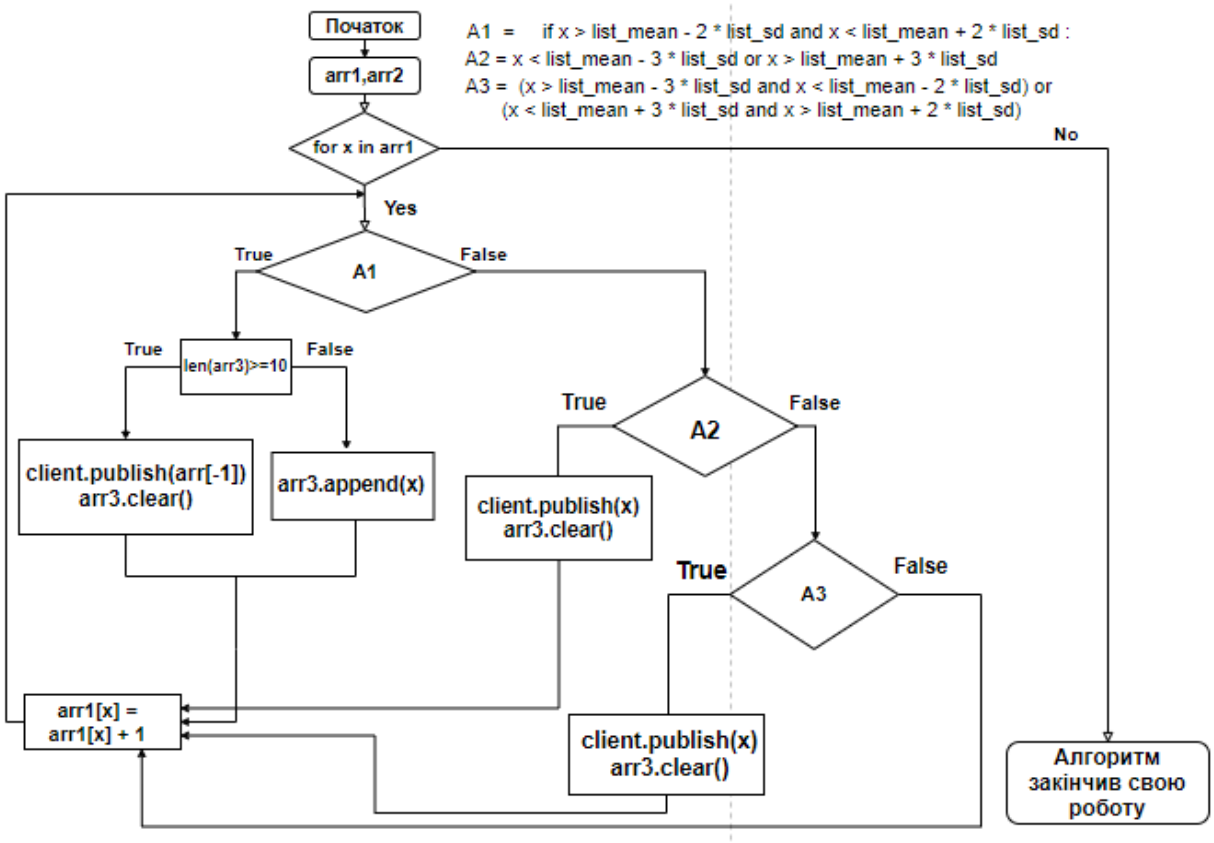


Рисунок 6 - Блок-схема алгоритму мінімізації

Для опису дослідницької моделі було побудовано блок-схему алгоритму.

Реалізація алгоритму. Принцип роботи полягає в перевірці кожного значення яке він отримує на аномальність та типовість. За допомогою алгоритму середньоквадратичних відхилень ми знаходимо середньоарифметичне та середньоквадратичне значення вибірки температури, далі алгоритм порівнює отримане значення з середньоарифметичним і перевіряє на скільки середньоквадратичних відхилень воно зміщене. Відповідно, чим більше зміщення тим аномальнішим воно рахується .

Загальна реалізація моделі заснована на таких пунктах:

1. Симуляція роботи датчика за 1 хвилину. Задана вибірка з 60 значень, кожне значення якої перевіряється алгоритмом раз в секунду.
2. Робота алгоритму. Алгоритм містить в собі 3 умови для детектування аномалій і визначення типовості, а саме :
    1) якщо значення лежить в межах до 2 середньоквадратичних відхилень, виконується мінімізація трафіку, тобто алгоритм накопичує значення за 10 секунд і відправляє на сервер лише останнє значення (так користувач зможе дізнатись температуру, яка була в момент отримання повідомлення), але для охоплення всіх аспектів проблеми розроблено також і надсилання типової температури одним повідомленням (всі значення, які отримані за 10 секунд надсилаються на сервер);
    2) якщо значення лежить в межах від 2 до 3 середньоквадратичних відхилень, пристрій одразу надсилає на сервер повідомлення про підвищення або пониження температури відносно норми;
    3) якщо значення лежить в діапазоні більше 3 середноквадратичних відхилень, пристрій одразу надсилає на сервер повідомлення про аномальне значення;

При виконанні пунктів 2 та 3 метод очищує повністю масив типових значень і продовжує заповнювати його відповідно до умови про типовість або ж до детектування нової аномалії.

Передача даних реалізована за допомогою протокола MQTT і локального MQTT сервера. Всі дані, що надсилаються датчиком на сервер відправляють повідомлення на топік “university/sensor1”, після цього, щоб користувач мав змогу переглянути їх, використовується клієнт. Графічний клієнт MQTT-fx

надає змогу повноцінно взаємодіяти з сервером. Головною його особливістю є графічна складова де підключитись на сервер, підписатись на тему і отримати від нього інформацію можна всього в декілька кліків. Кожне повідомлення представлено назвою теми та рівнем QoS. Причому вивід інформації інтуїтивно зрозумілий. (программа MQTT fx наведена в додатку В)

Для спрощеного використання формул середньоквадратичного відхилення і середньоарифметичного значення в мові (відповідно до поставленої задачі) Python існує бібліотека NumPy (Numeric Python), яка надає додатковий функціонал для роботи з загальними математичними і числови операціями у вигляді пре-скомпільованих, швидких функцій.

2.5 Визначення критеріїв ефективності роботи алгоритму

Для вирішення задачі зменшення трафіку обрано алгоритм середньоквадратичних відхилень, на основі якого буде реалізовано алгоритм мінімізації. Щоб наочно продемонструвати роботу програми, спочатку необхідно симулювати роботу датчика за одну хвилину. Так, як алгоритм заснований на визначенні середньоквадратичного відхилення, то мінімізація трафіку буде відбуватись в залежності відхилення отриманого значення з датчика від середньоарифметичного значення вибірки температури (ця вибірка не має аномалій, а включає в себе типові значення вибірки).

В залежності від того на скільки середньоквадратичних відхилень значення відхилилось від середньоарифметичного значення, виникають 3 умови :

- Якщо значення лежать в межах до 1 середньоквадратичного відхилення, то вони вважаються типовими, тобто до них і застосовується мінімізація.
- Якщо значення лежать в межах від 2 до 3 середньоквадратичних відхилень, то вони мають характер аномалії, внаслідок чого їх необхідно

відправити негайно на сервер, для повідомлення про аномальність (підвищення або пониження температури).

- Якщо значення лежать вище 3 середньоквадратичних відхилень, то вони вважаються аномальними, повідомлення одразу відправляється на сервер з поміткою про значне відхилення від норми.

Середньоквадратичне відхилення та середньоарифметичне значення обчислюється для вибірки за формулами:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n},$$

де $\bar{X}$ - середньоарифметичне значення;

x_i – і-ий елемент вибірки;

n – обсяг вибірки.

$$S = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n}},$$

де S – середньоквадратичне значення;

$\bar{X}$ – середньоарифметичне значення вибірки;

x_i – і-ий елемент вибірки;

n – обсяг вибірки.

Так як для початку роботи алгоритму середньоквадратичних відхилень необхідна вибірка температури для обчислення середньоарифметичного та середньквадратичного значень, то було введено можливість введення

користувачем температури та відхилення за замовчуванням, завдяки чому після заповнення масиву значеннями температури котрі відповідно до заданих параметрів не є аномальними, алгоритм матиме змогу занайти необхідні значення середньоарифметичного значення і середньоквадратичного відхилення та працювати вже самостійно.

Використання середньоквадратичного відхилення дозволяє розробити адаптивний алгоритм, який зможе підлаштуватись до будь-якого середовища за допомогою зміни значення параметрів для сигми, що є його особливістю внаслідок чого зменшуються витрати ресурсів.

2.6 Дослідження роботи алгоритму за результатами моделювання

Для демонстрації мінімізації даних було побудовано 3 графіки, які наочно показують надсилання даних до мінімізації і після неї.

Рисунок 7 - Графік роботи програми в першому випадку

В першому випадку, коли алгоритм не застосовано, сервер буде отримувати 60 повідомлень за хвилину, тобто кожну секунду пристрій надсилає значення температури. Внаслідок того, що алгоритм середньоквадратичних відхилень не використовується, то користувач не буде повідомлений про

аномальні значення, а отже виникають і інші проблеми, пов’язані з правильністю роботи датчика, інформативністю і корисністю повідомлень.

Рисунок 8 - Графік роботи програми в другому випадку

У другому випадку коли було використано алгоритм мінімізації, можна побачити, що датчик не отримував повідомлення про температуру перші 10 секунд, після цього надійшло повідомлення про останню температуру, яка була на датчику, потім через декілька секунд надійшло повідомлення про те, що помічено аномальне значення, а отже повідомлення прийшло одразу. І так далі алгоритм надсилав мінімізовану інформацію з інтервалом, а помітивши аномалію повідомляв користувача про загрозу.

Рисунок 9 - Графік роботи програми в третьому випадку

В третьому випадку дані за інтервал часу надсилаються єдиним повідомленням, яке містить в собі всі значення за 10 секунд (m1,m2,m3,m4 – масиви з значеннями температури), завдяки цьому користувач має змогу переглянути всі типові дані, що були знайдені за цей період на відміну від 2 випадку.

Підводячи підсумки продемоснтрованих кроків вирішення поставленої задачі, ми маємо можливість спостерігати, які дані були отримані брокером в кожному з випадків, тобто надсилання останнього значення з датчика, надсилання відповідних значень температури масивом та детектування аномальних значень температури. (Додаток В)

Взявши до уваги реалізацію цих трьох випадків та об’єму даних, можна зробити такий висновок на основі результатів, які продемонстровані на графіках та в програмі Wireshark, завдяки чому ми маємо можливість перехопити пакети і детальніше проаналізувати результати роботи розробленого алгоритму мінімізації та переглянути, що містять в собі пакети котрі передаються через протокол MQTT: (Додаток Г)

1. В першому випадку алгоритм надсилав значення кожну секунду, внаслідок чого об’єм даних на шляху сенсор – брокер – клієнт склав 18240 байт та 120 повідомлень.(Додаток Г. 1)
2. В другому випадку проведено мінімізацію внаслідок чого за 10 секунд надсилалось одне повідомлення з отриманим останнім значенням, завдяки чому об’єм даних на шляху сенсор – брокер – клієнт склав 2882 байт та 16 повідомлень.(Додаток Г. 2)
3. В третьому випадку на відміну від другого дані надсилались як масив, що складається з нормальних значень за 10 секунд. Відносно випадку 2, об’єм повідомлення незначно зріс, але клієнт матиме можливість знати про кожне значення температури за інтервал часу. Об’єм даних на шляху сенсор – брокер – клієнт склав 3186 байт та 16 повідомлень.(Додаток Г. 3)

Порівнявши отримані результати повідомлень, можна зробити висновок про мінімізацію даних за 10 секунд в процентному відношенні :

- Об’єм даних 18240 байт, 120 повідомлень.
- Об’єм даних 2882 байт, 16 повідомлень (зменшення трафіку на 84,2 %)
- Об’єм даних (зменшення трафіку на 82,6%)

Важливим уточненням до вирахуваних даних є те, що повідомлення спочатку надсилається від підписаного сенсора на брокер, а потім від брокера до підписаного клієнта, тобто завдяки зменшенню кількості повідомлень при роботі методу, було мінімізовано дані, що надходять від сенсора до брокера і від брокера до клієнта.

ВИСНОВКИ

Розглянуті тенденції та проблеми IoT спонукають до розробки методів мінімізації трафіку.

Встановлення пріоритету даних дозволяє забезпечити мінімізацію трафіку. Статистичні алгоритми придатні для цієї задачі. Перевагою їх є можливість виконання при обмежених обчислювальних ресурсах та простота реалізації, що є важливим фактором для мереж IoT.

Одним з статистичних алгоритмів є алгоритм середньоквадратичних відхилень. Він придатний для виконання задачі та поставленим вимогам.

На основі вирахуваних даних алгоритмом мінімізації трафіку можна зробити висновки про ефективність його роботи. На основі 2 випадків мінімізація трафіку пройшла успішно, що можна спостерігати завдяки зменшенню кількості повідомлень і об’єму даних, що в процентному відношенню складає - в другому випадку - 84.2 %, а в третьому - 82.6 %.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Maciej Kranz. Building the Internet of Things: Implement New Business Models, Disrupt Competitors, Transform Your Industry [Текст] : книга / Maciej Kranz. 2016. – 272 с. : ISBN 978-1-119-28566-3.
2. Bruce Sinclair. IoT Inc: How Your Company Can Use the Internet of Things to Win in the Outcome Economy [Текст] : книга / Bruce Sinclair. 2017. – 304с. : ASIN B071DZZRQS.
3. Development of network anomaly detection system architecture [Text] / Levonevskiy D.K // Bull. / Modern information technologies. – 2014. – Vol. 56, № 3. – P. 108 – 114. – ISSN 1814-1196 (онлайн).
4. Analysis and Classification of Methods for Network Attack Detection [Текст] / Branitskiy A.A. // Bull. / Proceedings of SPIIRAS. – 2016. – Vol. 45, № 2. – P. 207 – 244. – ISSN 2078-9599 (онлайн).
5. Santoyo S. A Brief Overview of Outlier Detection Techniques [Електронний ресурс] / Sergio Santoyo. – 2017. – Режим доступу до ресурсу: https://towardsdatascience.com/a-brief-overview-of-outlier-detection-techniques-1e0b2c19e561.
6. Сталингс У. Интернет вещей: сетевая архитектура и архитектура безопасности [Електронний ресурс] / Уильям Сталингс. – 2017. – Режим доступу до ресурсу: http://internetinside.ru/internet-veshhey-setevaya-arkhitektura-i/.
7. Дьяконов А. Поиск аномалий [Електронний ресурс] / Александр Дьяконов. – 2017. – Режим доступу до ресурсу: https://dyakonov.org/2017/04/19/%D0%BF%D0%BE%D0%B8%D1%81%D0%BA-

%D0%B0%D0%BD%D0%BE%D0%BC%D0%B0%D0%BB%D0%B8%D0%B9-anomaly-detection/.

8. Correa A. Benefits of Anomaly Detection Using Isolation Forests [Електронний ресурс] / Alejandro Correa. – 2016. – Режим доступу до ресурсу: https://blog.easysol.net/using-isolation-forests-anamoly-detection/.
9. Seif G. The 5 Basic Statistics Concepts Data Scientists Need to Know [Електронний ресурс] / George Seif. – 2018. – Режим доступу до ресурсу: https://nuancesprog.ru/p/2479/.
10. Mehedi H. Top 15 Standard IoT Protocols That You Must Know About [Електронний ресурс] / Hasan Mehedi. – 2018. – Режим доступу до ресурсу: https://www.ubuntupit.com/top-15-standard-iot-protocols-that-you-must-know-about/.

ДОДАТКИ

Додаток А

Програмний код розробленого алгоритму мінімізації трафіку

```
import numpy

import paho.mqtt.client as mqtt #імпортуємо MQTT-клієнт

import time

#import Adafruit_DHT


def on_log(client, userdata, level, buf):

    print("log: "+buf)

def on_connect(client, userdata, flags, rc):

    if rc==0:

        print("connected OK")

    else:

        print("Bad connection Returned code =",rc)

def on_disconnect(client, userdata, flags, rc=0):

    print("DisConnected result code "+str(rc))

broker="127.0.0.1" #"10.25.210.11"

client = mqtt.Client() #створюємо новий instance

client.on_disconnect=on_disconnect

client.on_log=on_log

print("Підключення до брокера",broker)

arr1 = []

arr2 = []

arr3 = []

arr4 = []
```

```
arr5 = []

arr = []

arr4.clear()

arr3.clear()

list_mean1 = 23

list_sd1 = 3

client.connect(broker) #підключення до брокера

client.loop_start() #loop begin

arr2 = []

#Вибірка температури датчика

arr1 =
[21,30,25,21,21,24,20,28,27,26,25,26,80,24,21,22,23,21,22,24,25,26,21,22,21,40,50,25,28,
31,20,21,22,23,25,25,25,26,21,23,25,27,27,27,28,25,26,26,22,21,20,19,21,20,22,22,21,23,2
4,25]

print("")

print("\n[Ініцалізація роботи датчика]")

client.publish("university/sensor1","[Розпочато ініціалізацію роботи датчика]")

#while True:

#    humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)

#    if humidity is not None and temperature is not None:

#        arr1.append(temperature)

        #Еталонна вибірка температури

arr =
[21,22,23,21,22,23,21,22,23,21,22,23,21,22,23,21,22,23,21,22,23,21,22,23,23,21,
22,23,21,22,23,23,21,22,23,21,22,23,23,21,22,23,21,22,23,23,21,22,23,21,22,23,23,21,22,2
3,21,22,23,23,21,22,23,21,22,23,23,21,22,23,21,22,23,23,21,22,23,21,22,23,23,21,22,23,21
,22,23,23,21,22,23,21,22,23,23,21,22,23,21,22,23,23,21,50,23,21,22,23]

#arr =
[21,25,21,27,27,28,25,21,20,17,30,25,20,21,25,25,28,23,20,21,20,25,27,30,25,20,19,17,21,
```

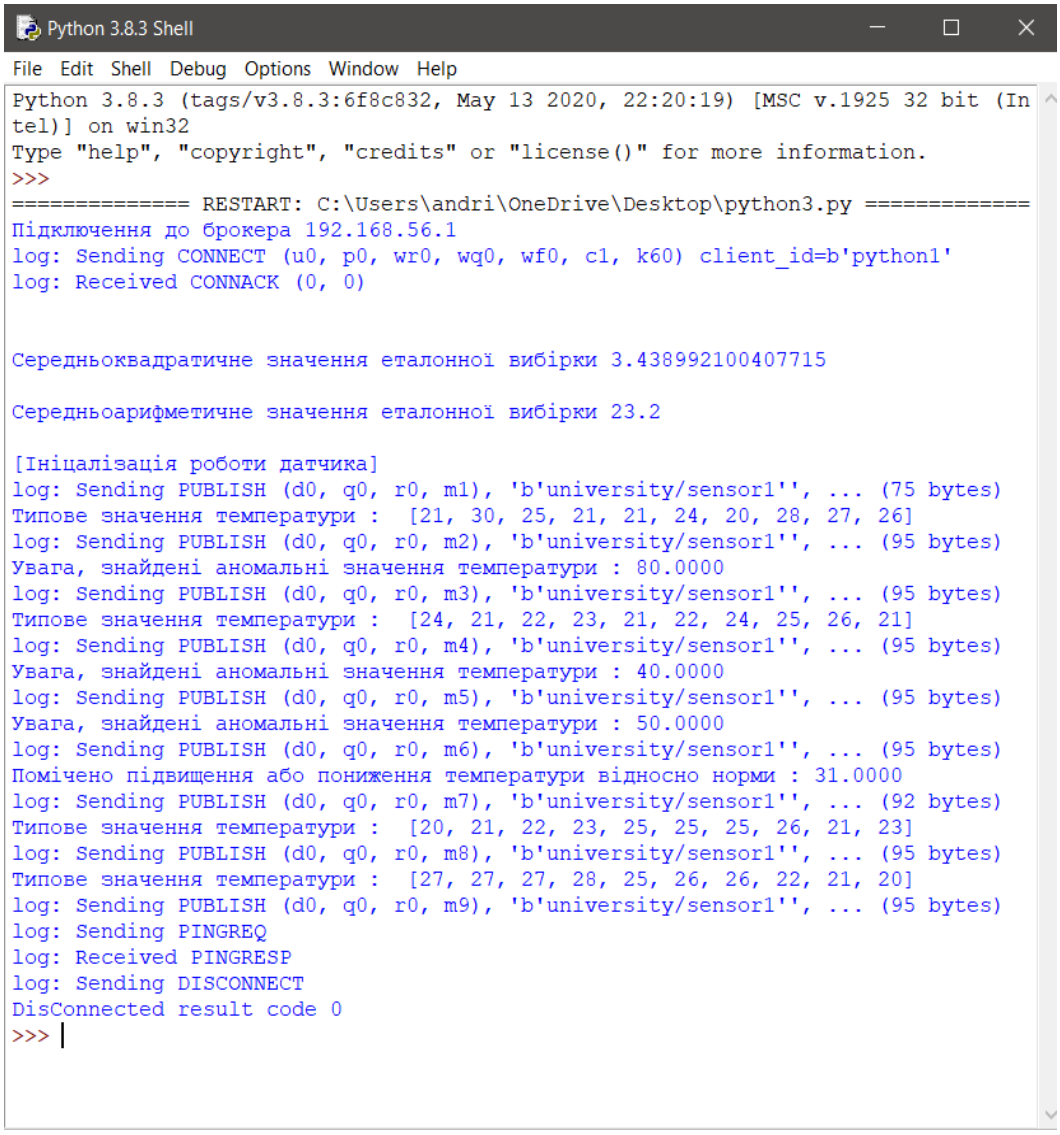
```
23,20,21,21,23,22,25,27,22,23,21,20,23,21,23,25,27,22,22,21,21,20,25,24,25,15,20,27,28,30,31]

#Отримуємо щосекунди значення температури з датчика

for x in arr:

#Перевіряємо значення температури на предмет типовості (отримання даних відбувається раз в 10 секунд)

    if x > list_mean1 - 2 * list_sd1 and x < list_mean1 + 2 * list_sd1 :

        if len(arr3) >= 10 :

            print("Типове значення температури : ",arr3[-1])

            client.publish("university/sensor1","Типове значення температури : %s"%str(arr3))

            arr3.clear()

        else :

            arr3.append(x)

            arr4.append(x)

            #Перевіряємо значення температури на предмет нетиповості (аномальності) (повідомлення надсилаються одразу)

    elif x < list_mean1 - 3 * list_sd1 or x > list_mean1 + 3 * list_sd1 :

        print("Увага, знайдені аномальні значення температури : %s"%x)

        client.publish("university/sensor1","Увага, знайдені аномальні значення температури : %s"%x)

    elif (x > list_mean1 - 3 * list_sd1 and x < list_mean1 - 2 * list_sd1) or (x < list_mean1 + 3 * list_sd1 and x > list_mean1 + 2 * list_sd1) :

        print("Помічено підвищення або пониження температури відносно норми : %s"%x)

        client.publish("university/sensor1","Помічено підвищення або пониження температури %s"%x)

    if len(arr4) == 60 :

        print("Amount of numbers : ",len(arr4))

        break
```

```
    time.sleep(1)
elements = numpy.array(arr)
#Знаходимо середньоарифметичне значення еталонної вибірки
list_mean = numpy.mean(elements, axis=0)
#Знаходимо середньоквадратичне значення еталонної вибірки
list_sd = numpy.std(elements, axis=0)
print("\nСередньоквадратичне значення еталонної вибірки",list_sd)
print("\nСередньоарифметичне значення еталонної вибірки",list_mean)
for x in arr:
    #Перевіряємо значення температури на предмет типовості (отримання даних відбувається раз в 10 секунд)
    if x > list_mean - 2 * list_sd and x < list_mean + 2 * list_sd :
        if len(arr5) >= 10 :
            print("Типове значення температури : ",arr5[-1])
            client.publish("university/sensor1","Типове значення температури : %s"%str(arr3))
            arr5.clear()
        else :
            arr5.append(x)
            #Перевіряємо значення температури на предмет нетиповості (аномальності) (повідомлення надсилаються одразу)
    elif x < list_mean - 3 * list_sd or x > list_mean + 3 * list_sd :
        print("Увага, знайдені аномальні значення температури : %s"%x)
        client.publish("university/sensor1","Увага, знайдені аномальні значення температури : %s"%x)
    elif (x > list_mean - 3 * list_sd and x < list_mean - 2 * list_sd) or (x < list_mean + 3 * list_sd and x > list_mean + 2 * list_sd) :
        print("Помічено підвищення або пониження температури відносно норми : %s"%x)
```

```
        client.publish("university/sensor1","Помічено підвищення або пониження
температури %s"%x)

    time.sleep(1)

time.sleep(4)

client.loop_stop() #Stop loop

client.disconnect() #disconnect
```

Додаток Б

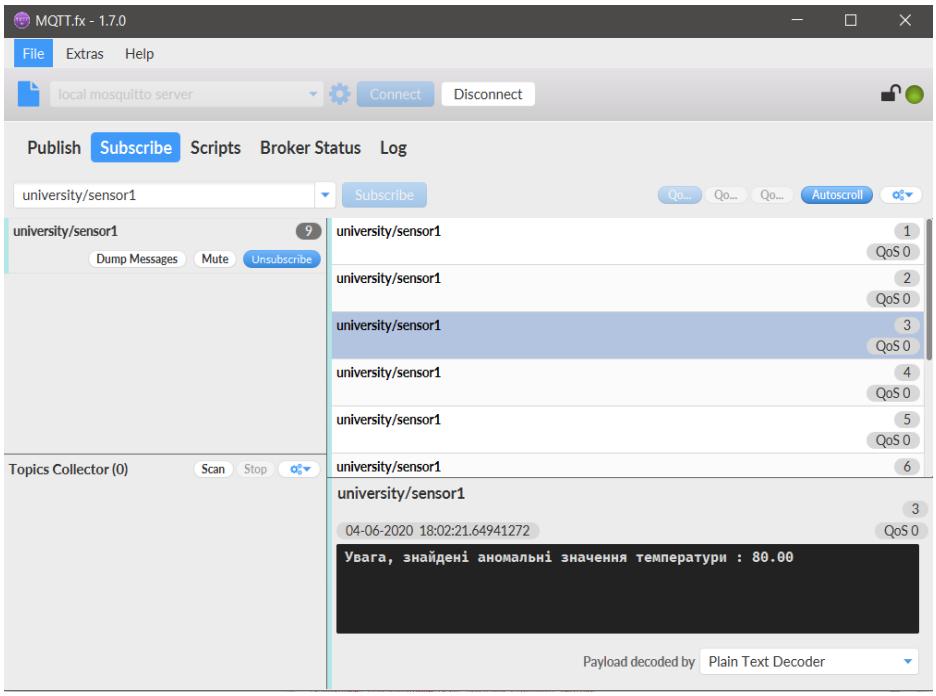
Демонстрація роботи методу мінімізації трафіку

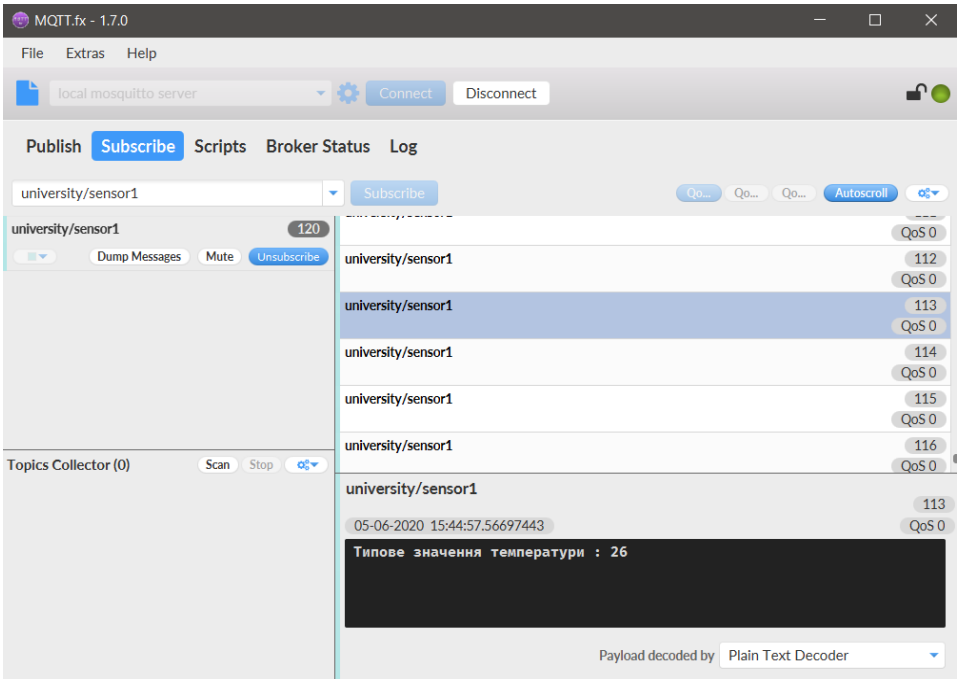


Додаток В

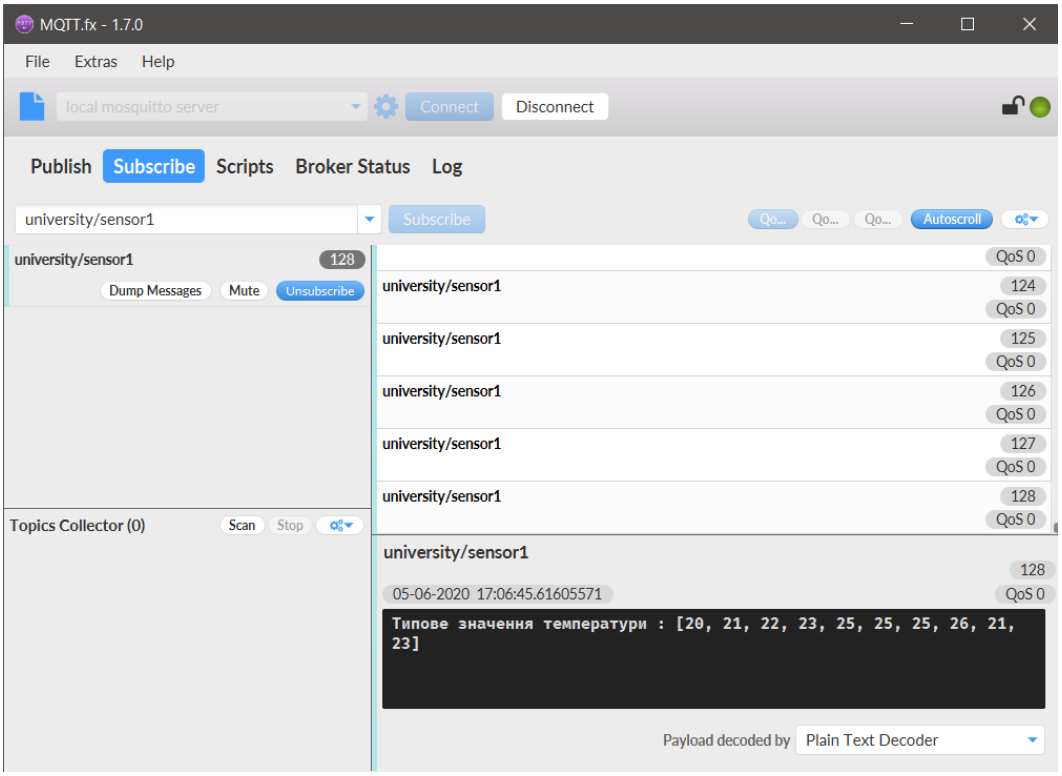
Результат роботи алгоритму на основі отриманих повідомлень клієнтом MQTT fx.

Отримані результати для другого випадку.





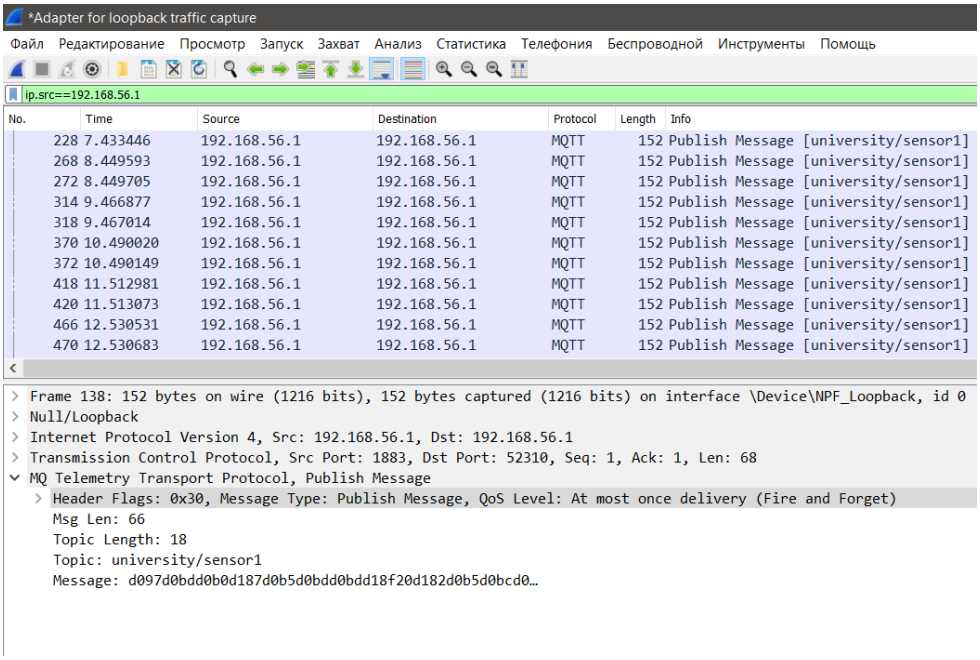
Отримані результати для третього випадку.



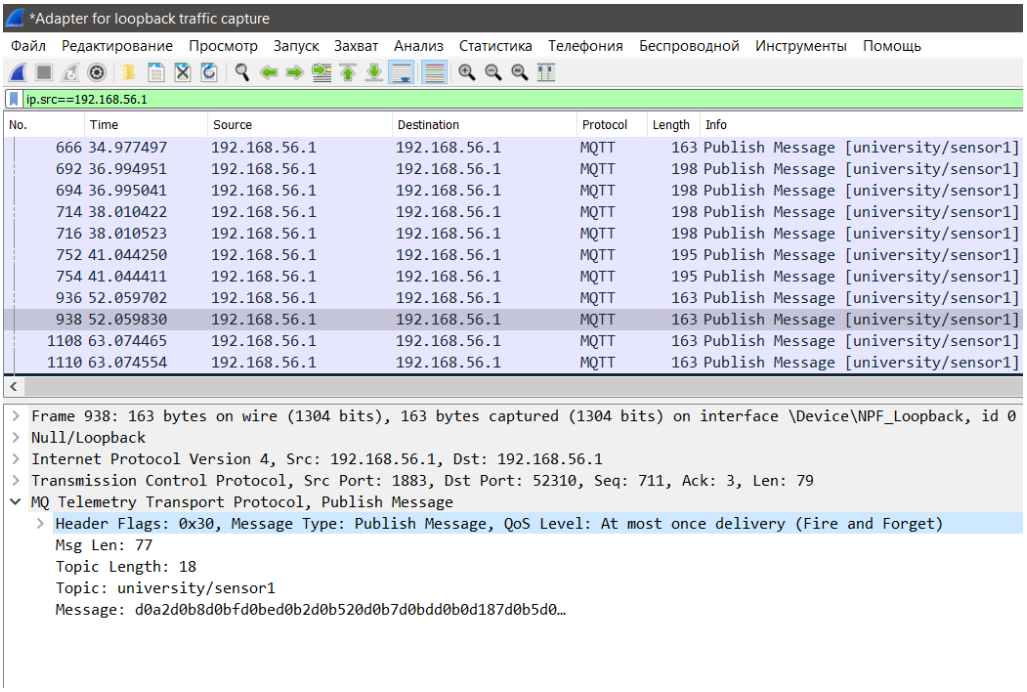

```
[andry@iot-hub-110 ~]$ mosquitto_sub -h 10.25.210.11 -t "university/sensor1"
[Розпочато ініціалізацію роботи датчика]
Типове значення температури : [20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 2
0.0, 20.0]
```

Додаток Г

1. Робота програми без реалізації алгоритму.



2. Робота програми на основі алгоритму мінімізації (повідомлення надсилається раз в 10 секунд).



3. Робота програми на основі алгоритму мінімізації.(повідомлення містить в собі всі значення за 10 секунд.)