Тема магістерської дисертації:

РЕФЛЕКСИВНІ ЗАСОБИ ОБ'ЄКТНОГО ВІДОБРАЖЕННЯ ДАНИХ ПО ПРОТОКОЛУ MODBUS

Виконав: магістрант НТУУ «КПІ імені Ігоря Сікорського»

ТЕФ, гр. ТІ-61м

Корнійчук М.А.

Керівник дипломної роботи:

к.т.н. Смаковський Д.С.

Актуальність

Взаємодія із супутниками та іншими пристроями які роблять певні заміри є дуже важливою задачею. Трудомістким є процес декодування та подальшого об’єктного відображення переданих даних з таких пристроїв по протоколу Modbus. Тому задача створення універсальної системи декодування та об’єктного відображення даних переданих по протоколу Modbus, а також пошук найшвидшого способу передачі цих даних в кожній конкретній ситуації є актуальною.

Мета роботи

Метою цієї роботи є створення алгоритмічної та програмної бази для декодування та подальшого об’єктного відображення даних переданих по протоколу Modbus з використанням рефлексивних засобів та удосконалення алгоритму передачі цих даних.

Задачі

Основні задачі цієї роботи полягають в:

- аналізі програмних засобів для отримання даних по протоколу Modbus;
- удосконаленні алгоритму передачі даних по протоколу Modbus;
- створенні функціоналу декодування та подальшого об’єктному відображення цих даних;
- розробці програмного забезпечення, яке буде реалізовувати цей новий функціонал.

Наукова новизна

Дослідивши протокол передачі даних Modbus й проаналізувавши існуючі бібліотеки для роботи з ним, було удосконалено алгоритм обміну даними з Modbus-пристроями і запропоновано об'єктне відображення отриманих даних, використовуючи рефлексивні засоби. Це значно зменшує кількість написання коду та удосконалює процес використання транзакцій, що в свою чергу скорочує витрати часу на отримання даних від пристрою.

Рефлексія

Рефлексія (в інформатиці) — це процес, під час якого комп'ютерна програма може відслідковувати і модифікувати власну структуру і поведінку під час виконання.

Modbus

Modbus – комунікаційний протокол, заснований на клієнт-серверній архітектурі.

Modbus-пристрої можуть зберігати інформацію в 4-х варіантах комірок:

- Discrete inputs: дискретні входи, тільки читання;
- Coils: дискретні виходи або внутрішні значення, тільки запис;
- Inputs: 16-бітні входи, тільки читання;
- Holding registers: 16-бітні виходи, читання та запис.

Огляд існуючих бібліотек для роботи з Modbus-пристроями

Таблиця порівняння проаналізованих бібліотек для роботи з протоколом Modbus за 3-ма критеріями, кожний з яких оцінений в балах від 1 до 5:

Назва бібліотеки	Функціональність	Детальність документації	Можливість пришвидшення передачі даних *
NModbus	4	5	–
Libmodbus	2	4	–
Java Modbus library	3	3	–
Jamod	4	5	–
jModbus	3	3	–
J2mod	3	3	–

* для кожної конкретної ситуації

Огляд існуючих варіантів зчитування даних з Modbus-пристрою

Наразі існує 2 варіанти зчитування даних з Modbus-пристрою:

1) кожний раз з пристрою вичитуються значення абсолютно з усіх регістрів в проміжку від одного необхідного регістру до наступного необхідного (включаючи дані з «зайвих» регістрів), максимальною довжиною в 125 регістрів;
2) кожна транзакція включає в себе інформацію про лише один числовий параметр, який може займати декілька регістрів, що розташовані послідовно один за одним, або розміщатися в одному регістрі.

Об’єктне відображення отриманих даних

Використовуючи розроблену бібліотеку для об’єктного відображення даних з Modbus-пристрою, програмісту необхідно всього лиш вказати анотацію над полями класу, та викликати відповідний метод бібліотеки, як показано на скріншотах нижче.

```
@ModbusFieldAnnotation(id = 1, port = 5555, inetAddress = "DESKTOP-KKD252G",
        reference = 1, length = 710000, k = 4, Vtrd = 500000)
private short status;

@ModbusFieldAnnotation(id = 1, port = 5555, inetAddress = "DESKTOP-KKD252G",
        reference = 10, length = 710000, k = 4, Vtrd = 500000)
private Short innerTemperature;

@ModbusFieldAnnotation(id = 1, port = 5555, inetAddress = "DESKTOP-KKD252G",
        reference = 2, length = 710000, k = 4, Vtrd = 500000)
private int voltage;
```

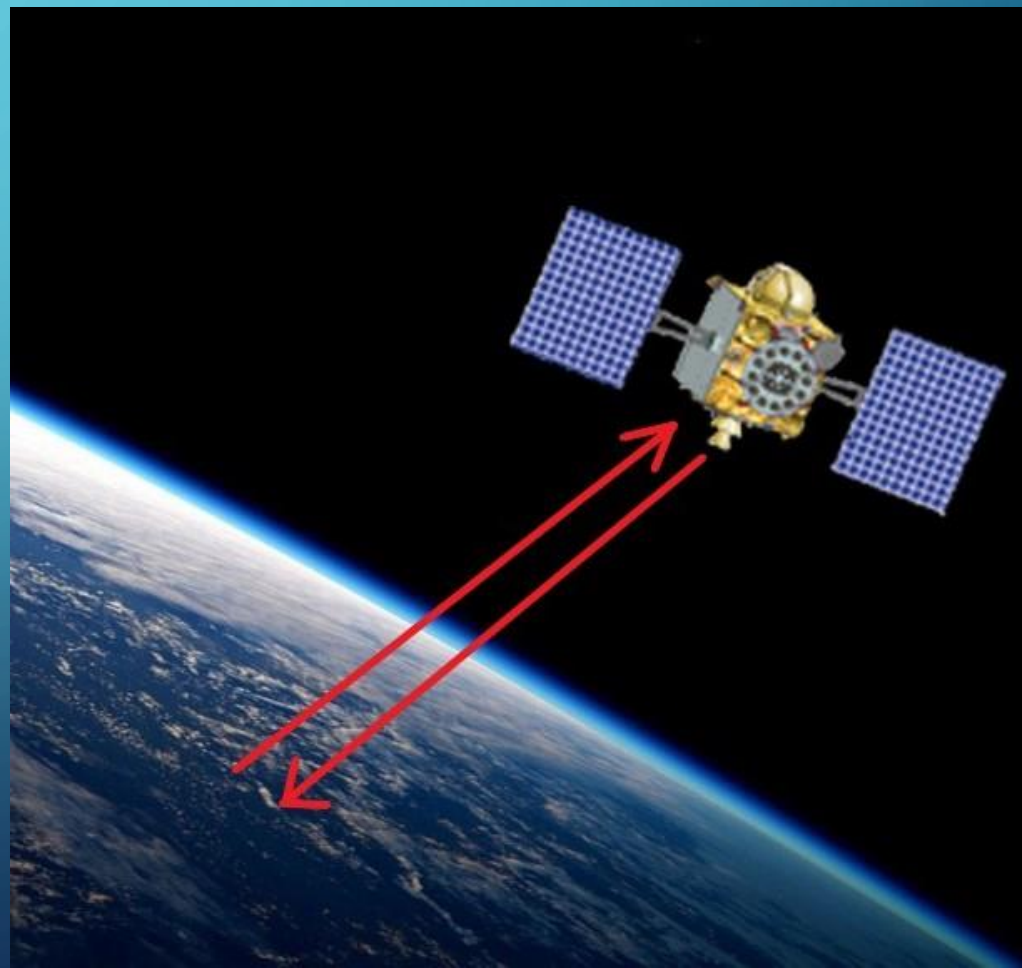
```
Device device = new Device();
OperationsWithAnnotation.initClassFieldsFromSlave(device);
```

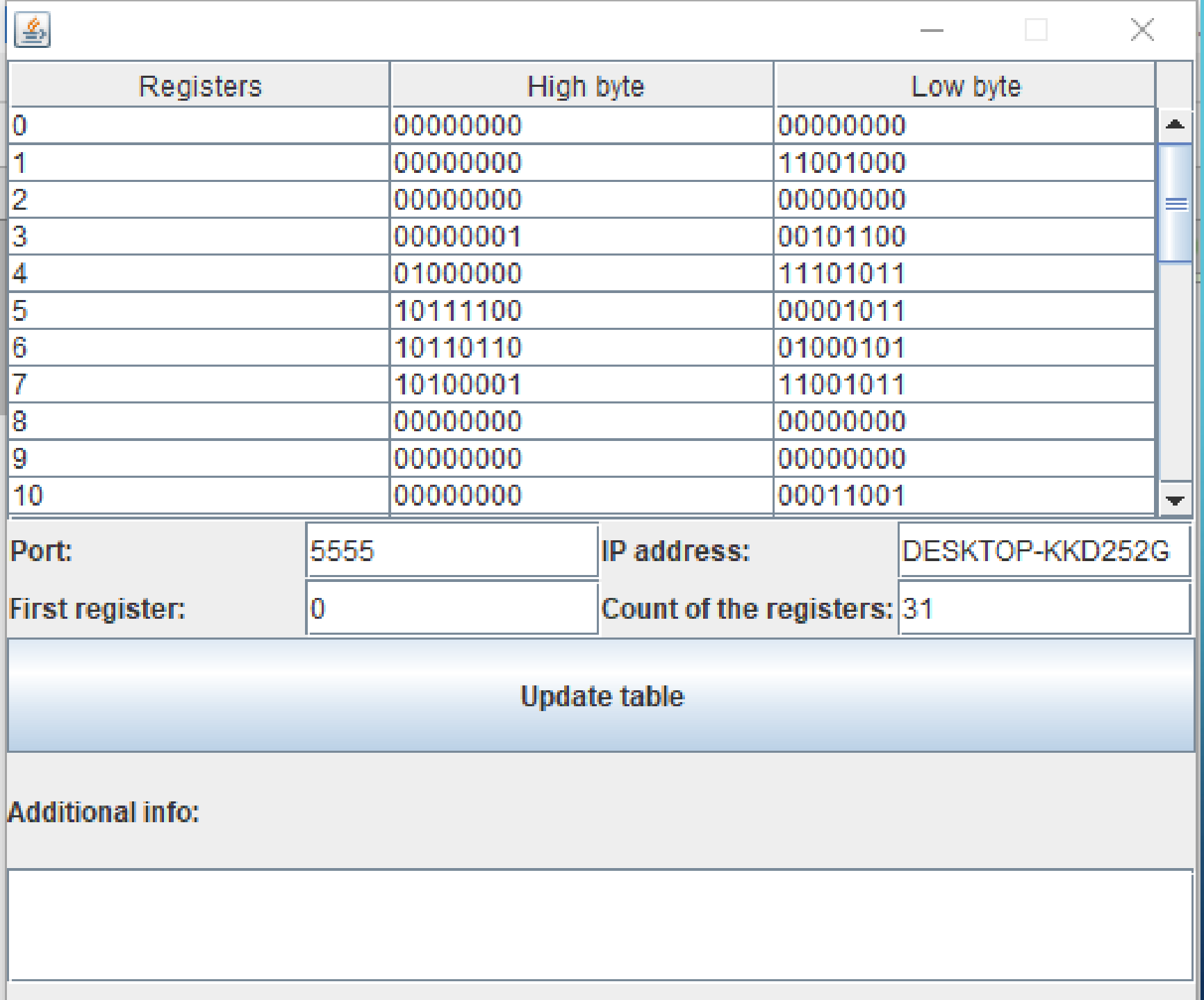
Особливість удосконаленого алгоритму

Алгоритм вираховує час, за який дані проходять відстань від Землі до супутника і назад. Назвемо цей час t_1. Потім алгоритм знаходить кількість регістрів (назвемо її числом N), час передачі даних з яких буде еквівалентний часу t_1. Далі, в проміжку необхідних 125 регістрів, алгоритм знаходить кількість «непотрібних» регістрів, розташованих один за одним, більшу за N. Наведемо формулу для обрахування N:

$$2 * t_{\text{тр.д.}} = k * \left(\frac{N * 16}{V_{\text{пер.д.}}} + t_{\text{пер.д.д.}} + t_{\text{тиш.}}\right)$$

де $t_{\text{тр.д.}}$ – час за який дані проходять відстань від Землі до супутника; k – статистичний показник, який вказує скільки транзакцій в середньому треба відправити щоб хоча б одна із них дійшла до супутника; $t_{\text{пер.д.}}$ – час за який дані з регістрів будуть передані антеною; $V_{\text{пер.д.}}$ – швидкість з якою антена передає дані; $t_{\text{пер.д.д.}}$ – час за який додаткові дані будуть передані антеною; $t_{\text{тиш.}}$ – час тиші в кінці транзакції.





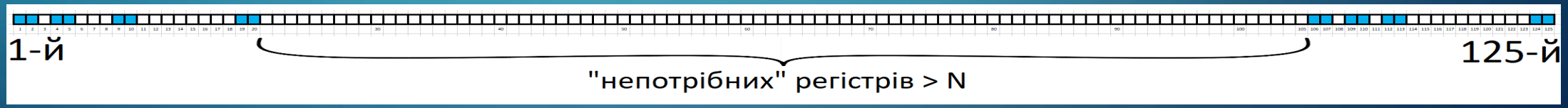
Особливості програмної реалізації

Програмний продукт був розроблений на мові програмування Java версії 1.8 з використанням бібліотек: Jamod, Lombok та Java Reflection API.

На картинці зліва – емулятор Modbus-пристрою.

Перший тестовий приклад

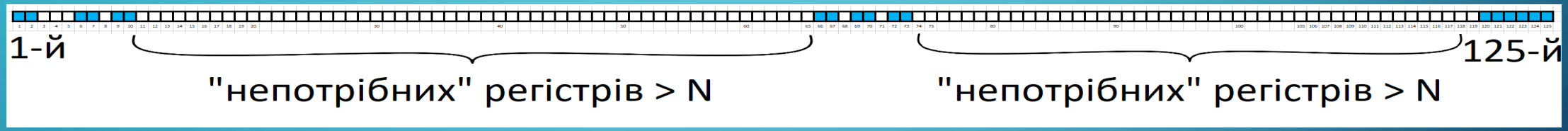
Нехай в тестових умовах було вирахувано, що в середньому до пристрою доходить без втрат даних лише кожна 4-а транзакція. За Modbus пристрій візьмемо супутник на орбіті 710 км. За швидкість передачі даних для супутника візьмемо 500 кбіт/с. Нехай маємо умовний відрізок в 125 регістрів, який зображено на рисунку нижче. Регістри, які необхідно зчитати з нього, позначені синім кольором. Нехай в цьому тестовому прикладі, нам необхідна інформація із 60-ти таких умовних відрізків.



Результат: алгоритм, який вичитує дані з усіх регістрів – 1.29 сек., алгоритм, який в кожній транзакції передає дані лише одного числового параметра – 5.39 сек., розроблений алгоритм – 0.73 сек.

Другий тестовий приклад

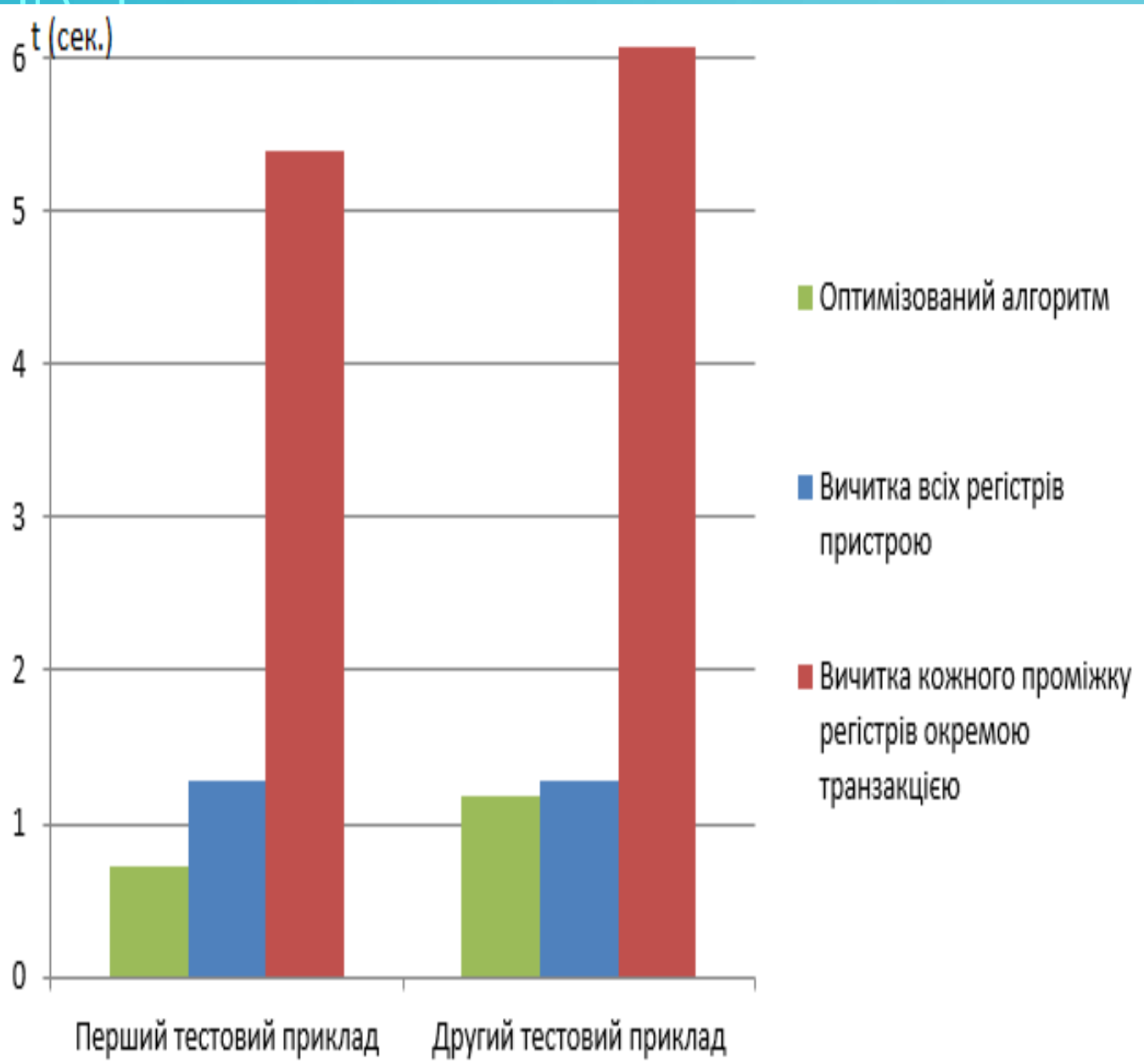
Нехай маємо ті ж початкові дані але необхідні регістри по-іншому розміщені в умовному відрізку довжиною в 125 регістрів, який зображений нижче. Регістри, які необхідно зчитати з нього, позначені синім кольором.



В цьому випадку маємо наступні результати:

- алгоритм, який вичитує дані з усіх регістрів – 1.29 сек.;
- алгоритм, який в кожній транзакції передає дані лише одного числового параметра – 6.07 сек.;
- розроблений алгоритм отримання даних – 1.17 сек.

Результати роботи програми



В 1-му тестовому прикладі удосконалений алгоритм (позначений зеленим) відпрацював на **44%** швидше ніж алгоритм який вичитує всі регістри (позначений синім) і на **87%** швидше ніж алгоритм який вичитує кожний регістр окремою транзакцією (позначено червоним).

Аналогічно в 2-му тестовому прикладі, удосконалений алгоритм відпрацював на **9%** швидше ніж «синій» алгоритм і на **81%** швидше ніж «червоний».

Висновки

Отже:

- був досліджений протокол передачі даних Modbus;
- були проаналізовані існуючі бібліотеки для роботи з ним;
- була створена алгоритмічна та програмна база для об’єктного відображення даних переданих по протоколу Modbus рефлексивними засобами;
- був удосконалений алгоритм передачі даних по протоколу Modbus.

Розроблена бібліотека пропонує фунціонал об’єктного відображення отриманих з Modbus-пристрою даних, а також функціонал пошуку та використання найшвидшого способу передачі цих даних в кожній конкретній ситуації.

Дякую за увагу !