

**Шифр: 282028CANI**

**Arduino UNO, як аналізатор інтерфейсу CAN для подальшого опрацювання інформації.**

## ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1. ОСОБЛИВОСТІ ФУНКЦІОНУВАННЯ CAN ШИНИ АВТОМОБІЛЯ, ВИМІРЮВАННЯ ОСЦИЛОГРАМ ТА ВИКОРИСТАННЯ ЗАВОДСЬКИХ АНАЛІЗАТОРІВ .....	4
1.1. Загальна інформація про CAN .....	4
1.2 Осцилограми CAN комфорт та двигуна .....	5
2.2. Використання заводських CAN аналізаторів Kvaser та Jonat .....	7
РОЗДІЛ 2. РОЗРОБЛЕННЯ ВЛАСНОГО CAN АНАЛІЗАТОРА ТА ОПРАЦЮВАННЯ ПОВІДОМЛЕНЬ .....	8
2.1. Що таке Arduino?.....	8
2.2. CAN застосунок.....	9
2.3. Data Logger застосунок .....	9
2.4. Збирання CAN аналізатора .....	10
2.5. З'єднання CAN аналізатора з транспортним засобом .....	12
2.6. Режими роботи CAN аналізатора .....	13
3.7. Режим отримання повідомлень .....	15
2.7. Режим фільтрування повідомлень.....	16
2.8. Режим передачі повідомлень .....	18
3.10 Опрацювання повідомлень .....	19
3.11 RPM (кількість обертів колінчастого валу) .....	20
3.12. Швидкість транспортного засобу .....	24
ВИСНОВКИ .....	29
АНОТАЦІЯ.....	31

## ВСТУП

На даний момент сучасний автомобіль використовує велику кількість електронних систем, число яких постійно зростає. Для їх ефективної роботи, необхідний інтенсивний обмін даними та інформацією, при цьому вимоги до кількості та швидкості передачі даних стають ретельнішими та вищими [1].

Наприклад, для забезпечення необхідної стійкості руху автомобіля електронна система курсової стійкості повинна (ESP) повинна обмінюватись даними із системами керування двигуном та трансмісією.

Необхідність встановлення зв'язку між ними, вимагає об'єднати всі електронні блоки керування в одну мережу.

Традиційний метод забезпечення цього обміну даних через окремі канали передачі даних від одної системи до іншої, досяг межі своїх можливостей (рис. 1.1), складність розведення електропроводів та розміри електричних роз'ємів не дозволяють виконувати ефективний контроль. Крім цього, обмежена кількість контактів в роз'ємах, збільшує складність розроблення блоків керування.

Єдиним рішенням цієї проблеми, застосування спеціальних та сумісних з автомобілем послідовних шин передачі даних, серед яких в якості стандарту була вибрана Controller Area Network (CAN).

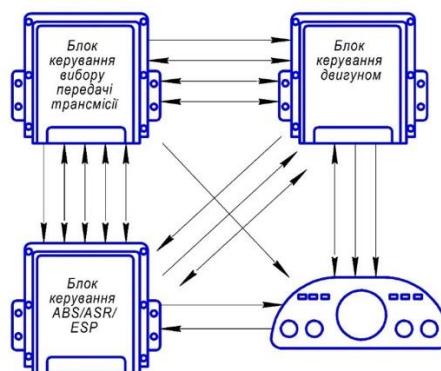


Рис. 1.1. Схема традиційної передачі даних

# РОЗДІЛ 1. ОСОБЛИВОСТІ ФУНКЦІОНУВАННЯ CAN ШИНИ АВТОМОБІЛЯ, ВИМІРЮВАННЯ ОСЦИЛОГРАМ ТА ВИКОРИСТАННЯ ЗАВОДСЬКИХ АНАЛІЗАТОРІВ

## 1.1. Загальна інформація про CAN

Інтерфейс CAN розроблений компанією Robert Bosch GmbH в середині 1980-х років, і знайшов застосування не тільки в автомобілебудуванні, а також в промисловій автоматизації, технологіях «розумного будинку».

Шина CAN (Controller Area Network), бортовий контролер зв'язку, є лінійною системною шиною (рис. 1.2), розробленою спеціально для використання на автомобілях, хоча вона і знайшла інші галузі застосування (побутова техніка).

Дані послідовно передаються по спільній шині. Всі блоки керування мають доступ до цієї шини. Через інтерфейс CAN блоки керування можуть обмінюватись даними. За допомогою об'єднання в одну спільну мережу, потрібна менша кількість проводів, тобто по одному каналі можна обмінюватись великою кількістю даних і багато разів зчитувати ці дані.

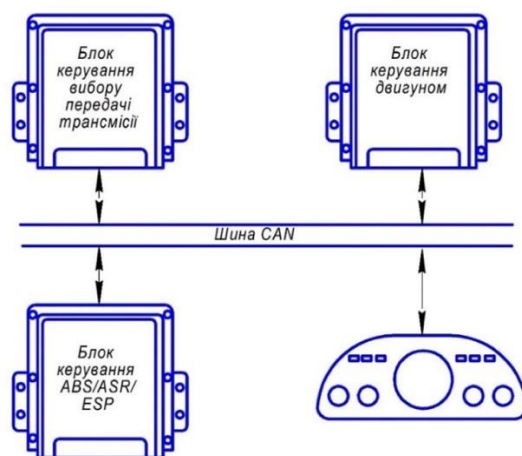


Рис.1.2. Лінійна структура шини даних CAN

Для першого ознайомлення з CAN шиною, було виміряні осцилограми CAN двигун/комфорт. Мета вимірювання, зрозуміти природу сигналів обидвох шин та проаналізувати їх амплітуди, частоти і періоди.

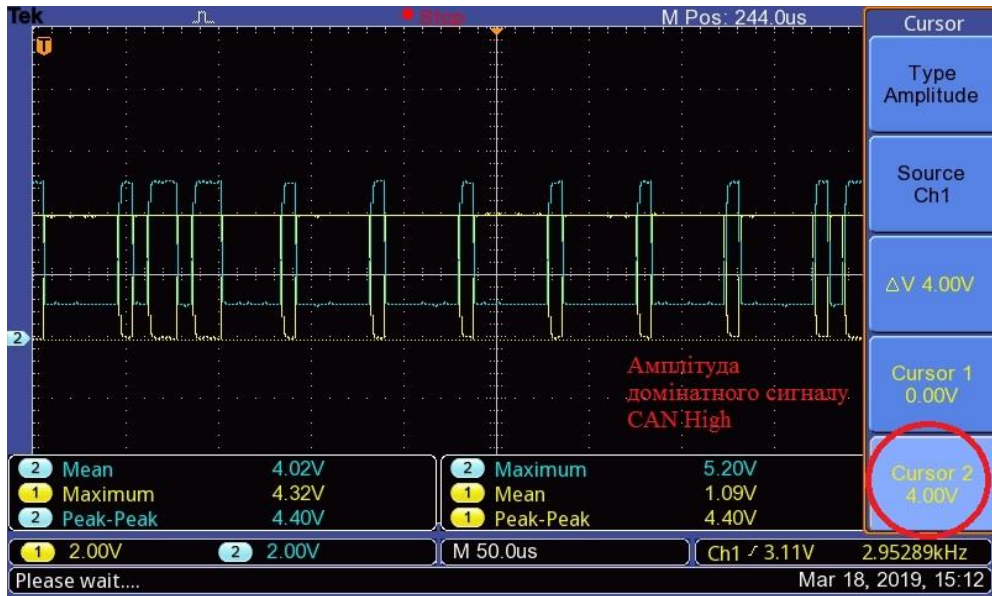
Для знайомства з кадрами повідомлень, вибрано заводські CAN аналізатори Kvaser та Jonat [5], кожен з яких має свої переваги та недоліки. Мета пристроїв: зчитувати, відправляти, фільтрувати повідомлення з CAN шини. Пристрої слугують проміжною ланкою між шиною обміну даних та комп'ютером.

На основі параметрів заводських аналізаторів, поставлено певні вимоги до майбутнього пристрою, а саме: можливість отримання; фільтрування; відправлення; записування повідомлень; відображення режимів пристрою, автономність. Перші дослідження проводилися на сучасному автомобільному демо-стенді Citroen C4. На фронтальній стороні стенду, знаходяться потенціометри, перемикачі, вимикачі, які імітують роботу сенсорів двигуна. Також розміщені панелі, для зручного підключення засобів вимірювання (рис. 2.2).

## **1.2 Осцилограми CAN комфорт та двигуна**

На рисунках нижче зображено осцилограми шини CAN комфорт. Амплітуди сигналів відповідають наступним значенням:

- в домінантному стані, напруга на провіднику High шини CAN збільшується до 3,6 В (рис.2.1);
- в рецесивному стані, напруга на провіднику High рівна 0 В, а на провіднику Low, приблизно 5 В;
- в домінантному стані, напруга на провіднику Low шини CAN зменшується приблизно до 1,4 В;



*Рис.1.3 Осцилограма шини CAN комфорт, на провіднику High в домінуючому стані*



*Рис.1.4 Стенд «Citroen C4»*

### 1.3. Використання заводських CAN аналізаторів Kvaser та Jonat

Наступним кроком для дослідження роботи CAN шини, використано аналізатор JONAT з програмним забезпеченням CanHacker. Перевага даної програми, можливість виводити повідомлення з одним і тим самим ідентифікатором в одну лінію.

Даними аналізатором вдалося, ідентифікувати декілька повідомлень:

CAN Комфорт (салону) 125kBit/s

Температура двигуна:

- ID488 3C 00 00 00 00 3C 00 00 відповідальний за 20°;
- ID488 77 00 00 00 00 3C 00 00 відповідальний за 50°;
- ID488 BB 00 00 00 00 3C 00 00 відповідальний за 120°;

*Примітка: підкреслений байт змінюється відповідно до температури двигуна.*

Тиск в системі кондиціонування:

- ID348 00 00 00 00 00 00 00 00 відповідальний за 0 bar;
- ID348 00 64 00 00 00 00 00 00 відповідальний за 15 bar;
- ID348 00 FF 00 00 00 00 00 00 00 відповідальний за 30 bar;

Селектор коробки передач:

- ID349 00 00 00 A7 00 00 00 00 відповідальний за R передачу;
- ID349 00 00 00 17 00 00 00 00 відповідальний за I передачу;
- ID349 00 00 00 26 00 00 00 00 відповідальний за II передачу;
- ID349 00 00 00 35 00 00 00 00 відповідальний за III передачу;
- ID349 00 00 00 44 00 00 00 00 відповідальний за IV передачу.

Дані повідомлення були виявлені візуальним шляхом, а саме: при задіяні того чи іншого перемикача, потенціометра, вимикача, відслідковувалися зміни байтів у повідомленні. Це вдалося зробити завдяки, розташуванні повідомлень з одним і тим самим ідентифікатором в одній лінії. Також можна застосовувати фільтри по ID, тобто у вікні програми будуть відображатися інформація тільки з однаковим ідентифікатором.

## РОЗДІЛ 2. РОЗРОБЛЕННЯ ВЛАСНОГО CAN АНАЛІЗАТОРА ТА ОПРАЦЮВАННЯ ПОВІДОМЛЕНЬ

### 2.1. Що таке Arduino?

Arduino Uno – широко використовувана плата, на базі мікроконтролера ATmega 328F (рис.3.1). Своєрідний конструктор, для любителів електроніки, який дозволяє без втручання паяльного обладнання, використовувати даний мікроконтролер. У своєму складі має: 14 цифрових входів/виходів, 6 з яких можуть виконувати функцію ШІМ сигналу (широтно-імпульсна модуляція), кварцовий резонатор на 16 МГц, 6 аналогових входів, роз'єм живлення, роз'єм для програмування (ICSP) і кнопка скидання (reset) [6].



*Рис.2.1. Arduino Uno*

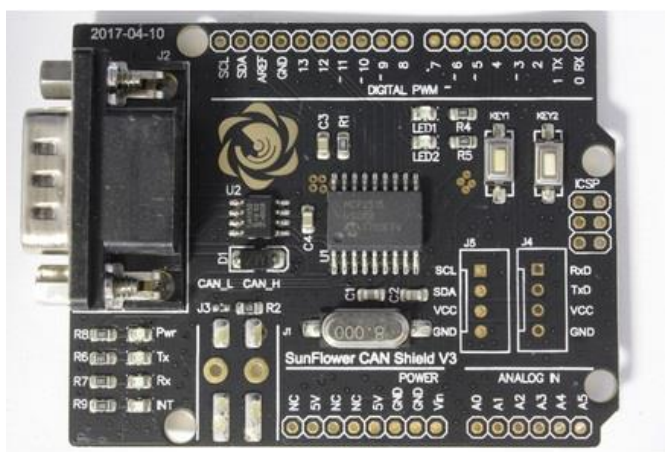


## 2.2. CAN застосунок

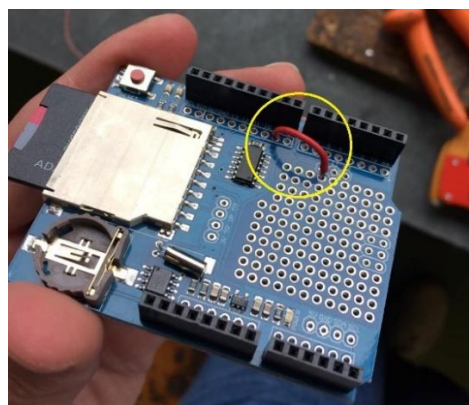
Для забезпечення зв'язку, між CAN шиною автомобіля, та Arduino, недостатньо одного мікроконтролера ATmega, потрібен проміжний пристрій, який би забезпечував спілкування з інтерфейсом, тому, вибрано CAN застосунок.

Даний застосунок надає Arduino можливості CAN шини, і дозволяє діагностувати або навіть внести певні корективи в автомобіль. Ця плата дозволяє опитувати електронні блоки керування. Для прикладу дізнатись інформацію про температуру охолоджувальної рідини, положення дросельної заслінки, швидкість транспортного засобу, обороти двигуна тощо [7].

Серцем пристрою слугує CAN контролер з інтерфейсом SPI - MCP2515, а також трансивер MCP2551 (рис.2.2). З'єднання з інтерфейсом транспортного засобу здійснюється за допомогою стандартного DB-9 роз'єму (COM порт), а також безпосередньо має контакти CAN, на самій платі.



*Рис. 2.2 CAN bus застосунок*



*Рис. 2.3 Data Logger застосунок*

## 2.3. Data Logger застосунок

Окрім, власне Arduino, CAN застосунок, було, також, використано Data Logger застосунок (рис 2.3).

Даний елемент використано з метою запису даних на карту пам'яті для подальшого імпорту в Excel та аналізу. Цей компонент, так само з'єднується з Arduino через SPI інтерфейс. Обидва застосунки Data Logger та CAN bus використовують однаковий вивід для передачі сигналу Slave Select (SS), (позначено жовтим кольором (рис.2.3)), тому даний вивід, перепаяно з контакту 10 на контакт 9, для узгодження пристроїв з мікроконтролером

## 2.4. Збирання CAN аналізатора

Після з'єднання усіх елементів між собою, отримано такий собі «бутерброд» з трьох основних плат: Arduino Uno, CAN та Data Logger застосунків (рис. 2.5).

Для з'єднання усіх вузлів і правильної їх роботи, потрібно запрограмувати мікроконтролер. Все програмування здійснювалось у середовищі Arduino IDE, на мові програмування Wiring, що є спрощеною підмножиною C/C++.

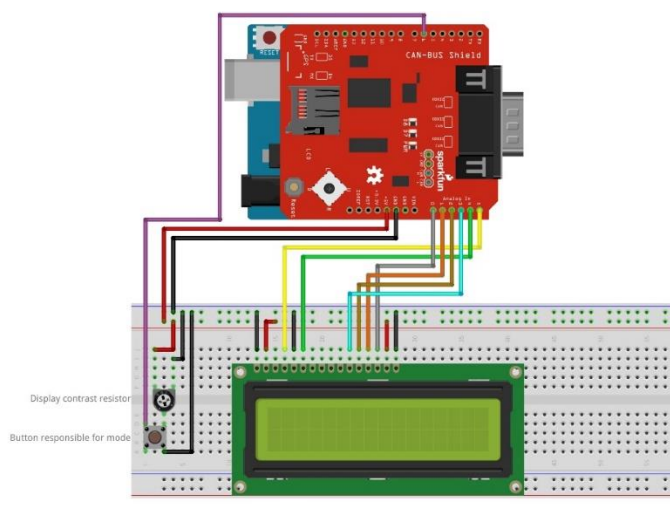


Рис. 2.4. Схема підключення

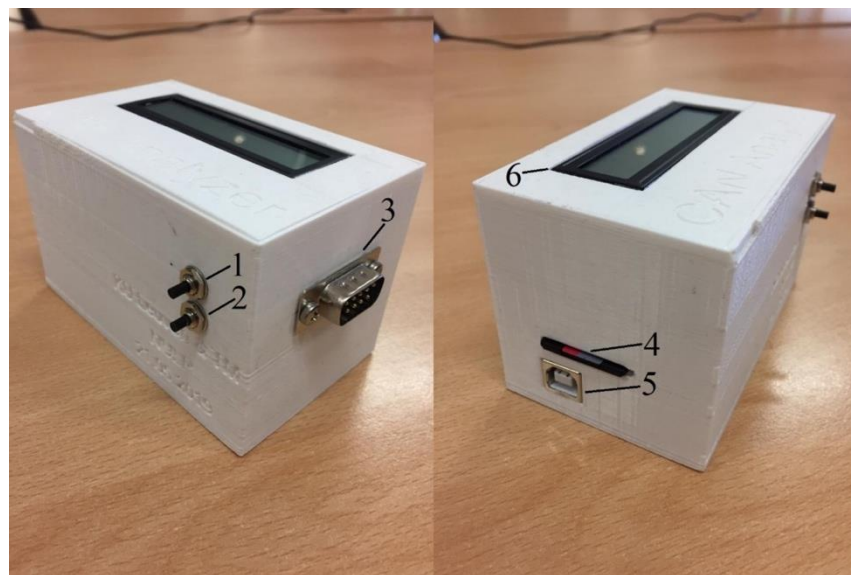


Рис.2.5. З'єднання основних компонентів

Базовий код та бібліотеки завантажено з середовища GitHub, яке в подальшому переписувалось під свої потреби, а саме застосування функції записування, відображення повідомлення та інших робочих функцій.

Після встановлення необхідних бібліотек була завантажена остання версія прошивки, та розпочались випробування вже безпосередньо на автомобілі.

Після багато чергових нюансів, які виникали в роботі, все ж таки було досягнуто оптимального рівня роботи, і було створено кінцевий продукт аналізатора (рис. 2.6). Всі елементи між собою, надійно, спаяні, а також роздрукований корпус на 3D принтері.



*Рис 2.6 Кінцевий продукт CAN аналізатора*

На рис. 2.6 позначено головні компоненти, а їх призначення наведено нижче:

1. кнопка відповідальна за зміну ID;
2. кнопка відповідальна за зміну швидкостей передачі даних;

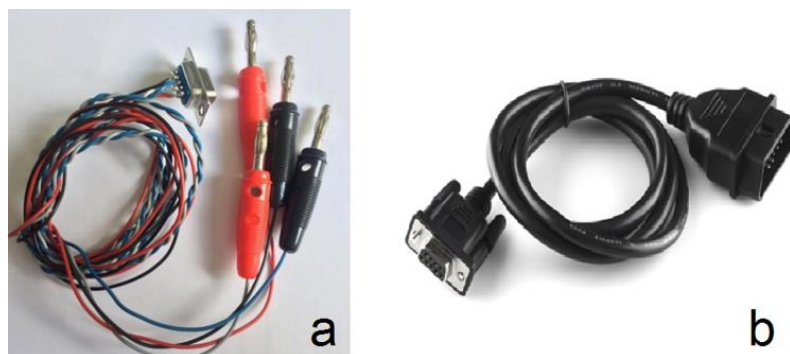
3. DB-9 роз'єм відповідальний для з'єднання аналізатора з транспортним засобом;
4. слот, відповідальний за ініціалізацію карти пам'яті;
5. USB порт (тип B), відповідальний за завантаження прошивки та живлення;
6. LCD дисплей, відповідальний за виведення інформації.

### 2.5. З'єднання CAN аналізатора з транспортним засобом

Існує декілька способів під'єднати CAN shield до лінії автомобіля: через OBD-II роз'єм, або безпосередньо в шину. Якщо у автомобілі задіяно тільки одну CAN шину, то підключення до OBD-II буде достатньо, якщо задіяно дві шини, з'єднання потрібно встановлювати як з діагностичним роз'ємом, який з'єднаний із шиною двигуна так і з шиною комфорту.

Для передачі даних, потрібно провідник, який би забезпечував безперебійний зв'язок між CAN застосунком та транспортним засобом. Ринок пропонує кабель (рис.2.7,(б)) одна сторона якого має роз'єм DB-9, а друга OBD-II.

Так, як не було можливості, придбати даний кабель, було спаяно другий, але з такими ж самими функціями провідник (рис.2.7 (а)). Де, червоний та чорний провідник відповідають за живлення, а синій та сірий відповідно за CAN сигнали.



*Рис.2.7. Загальний вигляд провідників: а - провідник, власного виробництва, б - заводський провідник*

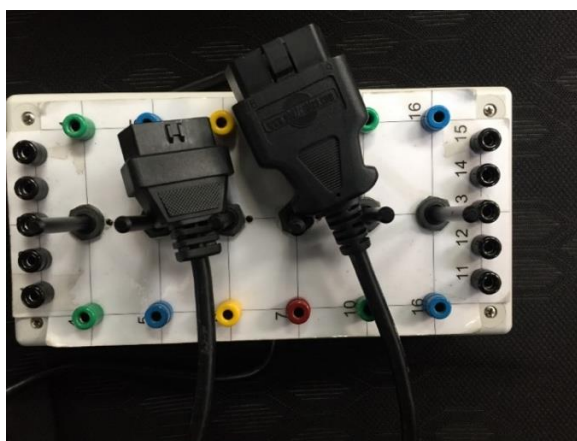
Також було використано адаптер (рис.2.8), який забезпечує надійний та комфортний доступ до OBD контактів, без їх пошкодження.

## 2.6. Режими роботи CAN аналізатора

Даний пристрій, може працювати в трьох основних режимах:

- режим отримання повідомлень;
- режим фільтрування повідомлень;
- режим передачі повідомлень.

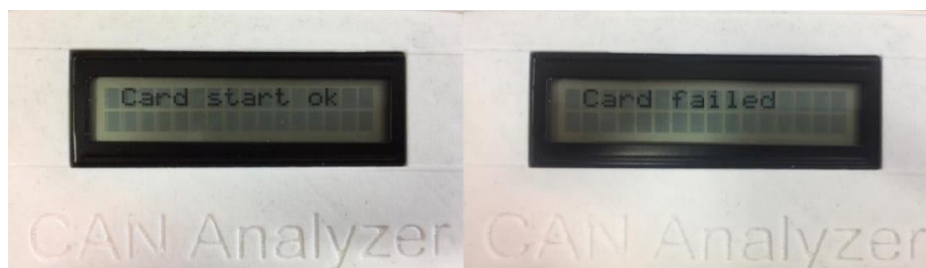
Ці режими активуються кнопками 1 та 2.



*Рис 2.8 Адаптер OBD-II*

Щоб розпочати роботу з CAN аналізатором, необхідно з'єднати пристрій з комп'ютером через USB порт та автомобілем через DB-9 порт. При необхідності записувати дані, також вставити карту пам'яті в слот ініціалізації 5.

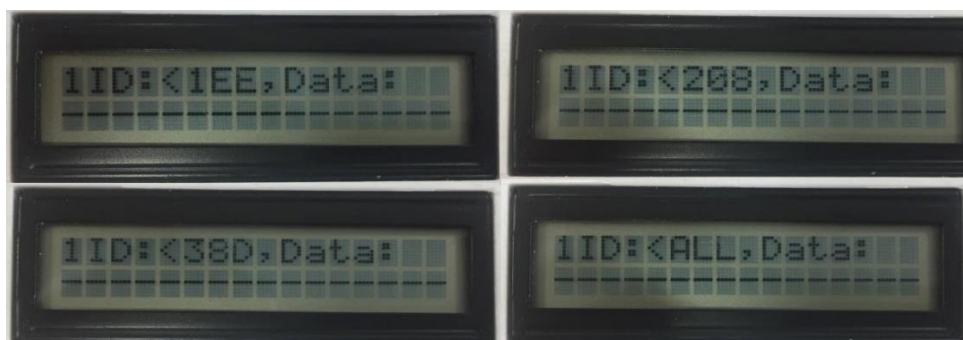
Після з'єднання аналізатора з комп'ютером, на дисплеї відобразиться перш за все інформація про ініціалізації карти пам'яті (рис.2.9).



*Рис. 2.9 Ініціалізація карти пам'яті*

Текст який слідує наступний, вказує інформацію про швидкість передачі даних, ID та вибраний тип режиму (приймання – передача), (рис.2.10).

Цифра «1» означає, що встановлена швидкість передачі даних 125 кБіт/с за замовчуванням. Після застосування кнопки «2» буде активована швидкість 500 кБіт/с. В результаті буде відображена цифра «5» замість «1» (рис.2.11).



*Рис. 2.10 Відображення різних ID*



*Рис.2.11 Швидкість передачі даних 500 кБіт/с*

Знак «<» означає «режим отримання повідомлень» активовано. Також може бути змінено ID натиснувши кнопку «1». Якщо натиснути кнопку «2» два рази «режим передачі повідомлень» буде активовано (рис.2.12).

Знак «>» означає «режим передачі повідомлень» активовано. Всі інші функції, такі ж самі, що для «режиму отримання повідомлень».



*Рис. 2.12 Режим передачі повідомлень*

В «режимі передачі повідомлень» повідомлення можуть відправлятися двома шляхами. Перший спосіб, утримання кнопки «1» протягом однієї секунди, в такому випадку відправляється одне повідомлення. Другий спосіб, утримання кнопки «1» протягом двох з половиною секунди, в такому випадку відправляється пакет повідомлень, тобто певна кількість повідомлень, яка раніше зазначена в прошивці.

## **2.7. Режим отримання повідомлень**

Режим CAN аналізатора забезпечує отримання і записування всього потоку даних з шини даних транспортного засобу. Для того, щоб активувати його, необхідно серед загального списку вибрати режим «ALL» (рис. 2.13) за допомогою кнопки «1» і відповідно швидкість кнопкою «2».



*Рис. 2.13 Режим отримання повідомлень*

Кожний тип транспортного засобу має різну швидкість передачі даних, тому перед застосування шукай інформацію про інтерфейс автомобіль. В

моєму випадку CAN комфорт – 125кБіт/с та CAN двигун -500кБіт/с. В прошивку, яка використовується завантажені дві швидкості відповідного до мого автомобіля. Ці дані можуть бути змінені шляхом заповнення інших значень у кодї програми (рис.2.14).

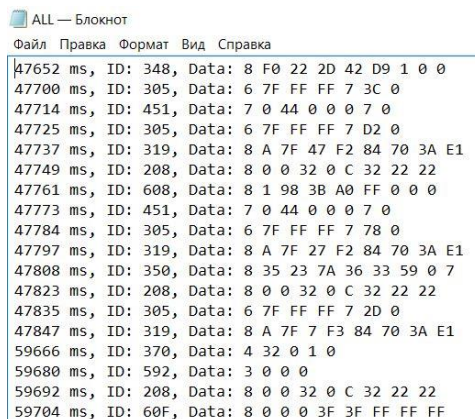
Даний пристрій підтримує три швидкості передачі даних:

- 125 кБіт/с;
- 250 кБіт/с;
- 500 кБіт/с.

```
if(Canbus.init(CANSPEED_500)) //Initialise MCP2515
```

*Рис.2.14 Швидкість передачі даних*

Під час застосування даного режиму весь потік даних відображається в Serial Monitor (SM) та паралельно записується на SD карту в <txt> формат для оф-лайн аналізу.



The screenshot shows a text editor window titled "ALL — Блокнот" with a menu bar containing "Файл", "Правка", "Формат", "Вид", and "Справка". The main text area displays a list of CAN bus messages in a structured format, including timestamps, IDs, and data bytes in hexadecimal. The messages are as follows:

Timestamp	ID	Data
47652 ms	348	8 F0 22 2D 42 D9 1 0 0
47700 ms	305	6 7F FF FF 7 3C 0
47714 ms	451	7 0 44 0 0 0 7 0
47725 ms	305	6 7F FF FF 7 D2 0
47737 ms	319	8 A 7F 47 F2 84 70 3A E1
47749 ms	208	8 0 0 32 0 C 32 22 22
47761 ms	608	8 1 98 3B A0 FF 0 0 0
47773 ms	451	7 0 44 0 0 0 7 0
47784 ms	305	6 7F FF FF 7 78 0
47797 ms	319	8 A 7F 27 F2 84 70 3A E1
47808 ms	350	8 35 23 7A 36 33 59 0 7
47823 ms	208	8 0 0 32 0 C 32 22 22
47835 ms	305	6 7F FF FF 7 2D 0
47847 ms	319	8 A 7F 7 F3 84 70 3A E1
59666 ms	370	4 32 0 1 0
59680 ms	592	3 0 0 0
59692 ms	208	8 0 0 32 0 C 32 22 22
59704 ms	60F	8 0 0 0 3F 3F FF FF FF

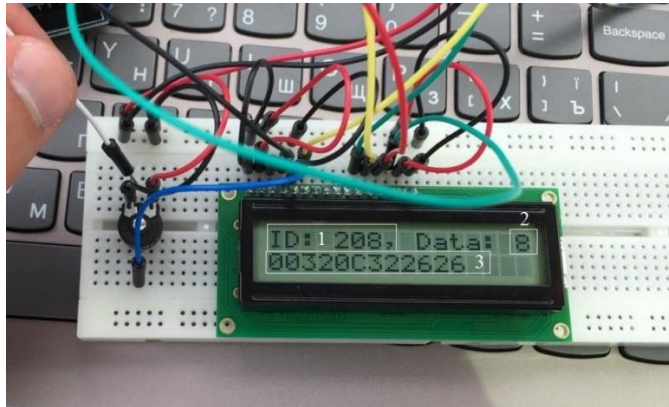
*Рис.2.15 Повідомлення збережені в <txt> формат під назвою «»ALL»*

## 2.8. Режим фільтрування повідомлень

Мета даного режиму відфільтрувати всі небажані повідомлення за ID. Тобто фільтрація відбувається, не за вмістом повідомлення. Один з найчастіше використаних режимів. Перед завантаженням прошивки в мікроконтролер



потрібно вказати всі бажані ID до яких буде застосовано фільтр в майбутньому. Також інформація, щодо повідомлень, відображається на дисплеї, де можна побачити основні частини.



*Рис.2.16 Відображення повідомлення з ID 208*

На (рис.2.16) відображено основні частини повідомлення, а саме: під позицією 1 – відображено ID, під позицією 2 – довжину повідомлення (кількість байтів), під позицією 3 – власне дані самого повідомлення.

Також даний кадр можна записати:

ID: 0x208, Data: 8 0x00 0x00 0x32 0x00 0x0C 0x32 0x26 0x26

Де символ «0x», означає запис у шістнадцятковій системі числення.

Кнопкою «1» є можливість перемикає ID. Для прикладу, на (рис 2.17) зображено рядок коду, який відповідає за фільтровані ідентифікатори.

```
int MESSAGE_ID[] = { 0x38D, 0x208 }; // Відслідковуване ID
```

*Рис.2.17 ID повідомлень*

Вся інформація виводиться в Serial Monitor та паралельно записується на карту пам'яті (рис. 2.18)

1EE	Дата изменения: 01.01.2000 0:00 Размер: 45,3 КБ
38D	Дата изменения: 01.01.2000 0:00 Размер: 8,03 КБ
44D	Дата изменения: 01.01.2000 0:00 Размер: 8,45 КБ
208	Дата изменения: 01.01.2000 0:00 Размер: 720 КБ
ALL	Дата изменения: 01.01.2000 0:00 Размер: 29,4 КБ

*Рис. 2.18 Відфільтроване повідомлення з ідентифікатором «208»*

## 2.9. Режим передачі повідомлень

Наступний логічний крок, відправити ці повідомлення назад в CAN лінію та активувати певний виконавчий механізм. На рис. 2.19 наведено приклад відправленої інформації

Повідомлення відправляється за допомогою кнопки «2», вище описаним методом (рис.2.12).

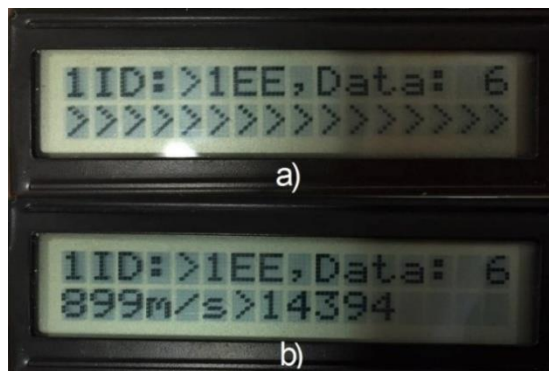
```

message.id = 0x1EE; //формат в шістнадцятковій системі числення(HEX)
message.header.rtr = 0;
message.header.length = 6; //формат в десятковій
message.data[0] = 0x00;
message.data[1] = 0x80;
message.data[2] = 0x01;
message.data[3] = 0x00; //формат HEX
message.data[4] = 0x00;
message.data[5] = 0x0b;
message.data[6] = 0x00;
message.data[7] = 0x00;

```

*Рис 2.19 Повідомлення, яке транслюється в шину*

Текст «899 m/s» (рис.2.20(б)) означає, що за одну секунду відправляється 899 повідомлень. Дане повідомлення, відповідає за рух водійського сидіння до передньої частини автомобіля. Так, як фотографія, відображає тільки зображення в статиці, рух сидіння можна переглянути за посиланням: <https://www.youtube.com/watch?v=aLc4r5ru0Ng>.



*Рис. 2.20. а) Відображення дисплею при відправленні одного повідомлення; б) відображення дисплею при відправленні пакету повідомлень*

Спосіб, дозволяє, активувати будь-який компонент автомобіля, який керується CAN шиною автомобіля. Потрібно лиш знати правильне повідомлення, і з правильним періодом часу транслювати його в лінію.

## **2.10 Опрацювання повідомлень**

Одним з найважливішим кроком дослідження було не просто навчитися отримувати повідомлення та відправляти їх назад в шину простим та дешевим методом, а також вміти їх розшифровувати, аналізувати та робити висновки. Без розшифрування повідомлень та їх аналізу, це лиш шматок цифрів в шістнадцятковій системі числення, які нічого не означають, для звичного користувача. Тому даний розділ, моїх тезисів буде, присвячено саме аналізу повідомлень.

З самого початку мого дослідження, вибрано модель CAN застосунок з можливістю під'єднувати SD карту пам'яті, для запису. Даний метод дозволяє, записати будь-яке ID повідомлення та, надалі, імпортувати ці файли до Excel, для аналізу.

Протягом мого дослідження було записано багато різних даних: інформацію від сенсору положення керма, педалі газу, педалі тормозу, селектору переключення передач, сенсора швидкості транспортного засобу, сенсора кількості обертів та інших. В цій частині роботи, буде звернена увага,

саме на сенсорі швидкості автомобіля, та сенсорі швидкості обертів двигуна.

Здебільшого всі електронні компоненти сучасного автомобіля, з'єднано з шиною даних, на даний момент це одні з найрозповсюджених шини: CAN та Flex Ray, а також локальні шини LIN та інші. Відповідно, якщо елементи підключено до цієї шини, можна отримати будь-яке повідомлення від сенсора, чи виконавчого механізму. Один з найважливішим кроком інтерпретувати ці кадри, для людського сприйняття, для прикладу швидкість записувати в км/год, кількість обертів в об/хв.

### **2.11 RPM (кількість обертів колінчастого валу)**

Для того, щоб дізнатись кількість обертів двигуна, потрібно перш за все ідентифікувати ID відповідальне за кількість обертів. Витративши багато часу, було з'ясовано, візуальним шляхом, що ID 208, відповідає з великою ймовірністю за RPM. Далі було застосовано фільтр, і підтверджено ще раз, що саме це ID є тим, що потрібно. Довжина повідомлень 8 байт (рис.2.21).

```
8269 ms, ID: 208, Data: 8 33 80 36 8 4C 32 36 36
```

*Рис.2.21. Повідомлення, кількості оборотів*

Як можна побачити на (рис.2.21), це всього лиш невідомий код, який нічого не означає, поки його не розшифрувати. Трохи, забігши наперед, варто написати, що дане повідомлення відповідає за 1633об/хв, колінчастого валу.

Записування потоку даних з цим ID декілька разів, дало можливість дізнатись, який байт за що відповідає. Для прикладу перших три байти 0x33, 0x80, 0x36, збільшуються відповідно до збільшення оборотів. Четвертий байт, 0x08 змінює своє значення відповідно до натискання педалі газу. П'ятий байт змінює своє значення з 0x4C на 0x4E при натисканні педалі тормозу. Шостий байт, 0x32 не змінює свого значення. Останні два байти змінюють свої

значення відповідно до збільшення кількості обертів, але у вузькому діапазоні. З лівої сторони (рис.2.21) можна бачити час, який дозволяє зробити процес аналізування більш легким та зручнішим.

Найбільшу увагу, було приділено першим двом байтам повідомленням, тому що їх значення при вимкненому двигуні рівні 0x00 та 0x00, що відповідає нульовій кількості обертів – двигун не запущений.

Щоб оперувати даними значеннями потрібно чітко встановити, яка кількість обертів відповідає за перші два байти повідомлення. Тому, було під'єднано діагностичний пристрій AutoLink 619 (рис.2.22), та в той самий час CAN Analyzer.



*Рис.2.22. Діагностичний пристрій AutoLink 619*

Було встановлено, що 683 об/хв є відповідальні за 0x15 та 0x78, а також інші значення кількості обертів та відповідні їм значення повідомлень.

**Значення кількості обертів та відповідні їм коди**

RPM	HEX формат	DEC формат	const. RPM	Середнє
683	0x15 та 0x78	5496	0.1242	0.125
1024	0x20 та 0x28	8232	0.1243	
1537	0x30 та 0xD8	12504	0.1229	
2532	0x4F та 0x60	20320	0.1246	
3447	0x68 та 0x58	26712	0.1290	

Наступним кроком, було переведено всі значення в десяткову систему числення (таблиця 2.1), та розраховано константу для RPM автомобіля Citroen C6 (таблиця 2.2).

$$\text{const. RPM} = \frac{\text{RPM}}{\text{DEC значення}} \quad (2.1)$$

В результаті було розраховано, середнє значення серед п'яти: 0,125 (1/8), таким чином, зараз є можливість розрахувати дійсну кількість обертів двигуна.

Щоб розрахувати дійсну кількість обертів потрібно перших два байти помножити на отриману константу (рис 2.23).

**Проміжні значення обертів**

Час, ms	208 Повідомлення	Перших два байти	DEC формат	Помножено на 0,125 [об/хв]
9704	00320E322727	00	0	0
9903	3B8810E322727	3B8	952	119
10632	1AC83D04E322A2A	1AC8	6856	857
20093	1EF84E84C324E4E	1EF8	7928	991
20571	2E9045A4C324545	2E90	11408	1426
23195	47D867224C326767	47D8	18392	2299

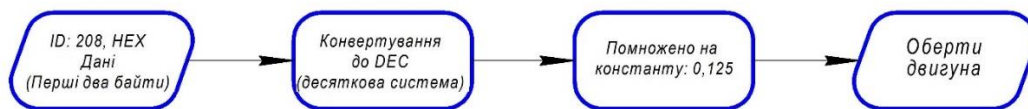


Рис. 2.23. Конвертування повідомлення з ID 208, в дійсну кількість обертів

Протягом тридцяти секунд, було записано дані з ID 208. Протягом цього часу кількість оборотів зростала, пропорційно натисканні педалі газу.

Серед великого масиву даних 1394 значень, було побудовано графік (рис.2.24), а також вибрано декілька проміжних точок, аби навести приклад побудови (таблиця 2.2).

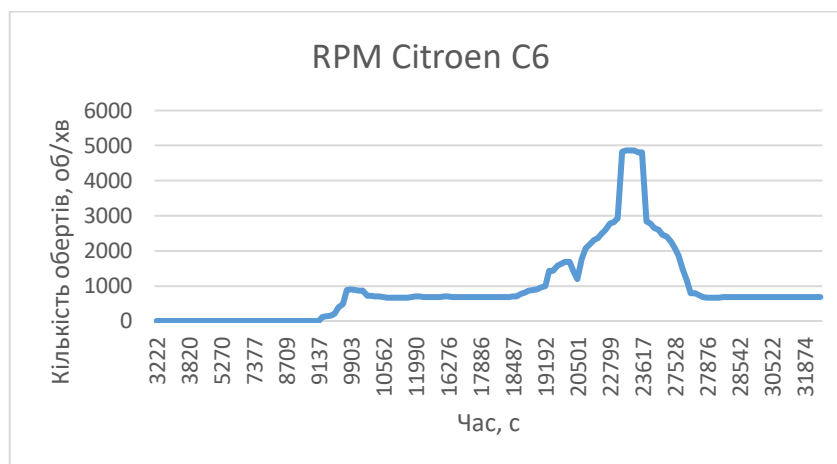


Рис.2.24. Графік зміни обертів

На графіку (рис. 2.24) відображено зміну кількості обертів, протягом тридцяти секунд. Як можна бачити, на інтервалі часу 0-9428 ms, кількість обертів рівна нулю, що свідчить про незапущений двигун, з 9428 ms, двигун запускався - розпочалося різке збільшення обертів близько 851 об/хв, інтервал часу з 10011 до 18154 ms відповідає марному ходу автомобіля і відповідно дорівнює 687 об/хв. Ділянка після марного ходу, відповідає, різкому збільшенню обертів, відповідно натискалася педаль газу, кількість обертів близько 5000 об/хв. З 23593 ms, відображено зменшення обертів, відповідно

тиск на педаль газу, було завершено, та кількість обертів знову стали відповідати марному ходу. Також слід сказати, що дані, які були використані для побудови діаграми є доволі точно, тому що зміна кількості обертів реєструвалася через кожні 40 мілісекунд. Цікавий момент відображено на 20571 ms, що відповідає помилці у шині даних, та зменшені обертів.

## 2.12. Швидкість транспортного засобу

Щоб розрахувати швидкість транспортного засобу, було використано подібний метод, як для RPM. Одна з найбільших проблем, забезпечити сталу швидкість протягом певного часу, що неможливо зробити біля будівлі університету. Тому автомобіль, було поставлено на підйомник (рис.2.25), і продовжено дослідження.



*Рис.2.25. Розташування автомобіля на підйомнику*

Перш за все, як і в попередньому методі, було визначено ID відповідальне за зміну швидкості транспортного засобу візуальним шляхом. Це повідомлення з ідентифікатором 38D та довжиною даних 5 байт: 0x38D 0x00 0x00 0x04 0x95 0xAF

Перші два байти змінюють свої значення відповідно до збільшення швидкості автомобіля. Протягом дослідження на підйомнику, встановлено



декілька значень швидкостей та записано відповідальні їм байти. В той самий час, відслідковувались три змінні (рис.2.26):

- Швидкість на панелі приладів в км/год;
- швидкість на діагностичному пристрої в миль/год;
- значення перших двох байтів.

Як показано на (рис.2.26) швидкість на панелі приладів дорівнює 9 км/год, в той час як швидкість на діагностичному пристрої рівна 4 миль/год та перші два байти дорівнюють 0x02 та 0x9B. Така ж сама операція, виконана для швидкостей 28; 33; 48; 55; 58; 60 км/год. Також, було звернено увагу на похибку спідометра, тому що різниця між даними на спідометрі та діагностичному пристрої є доволі значна (таблиця 3.3).

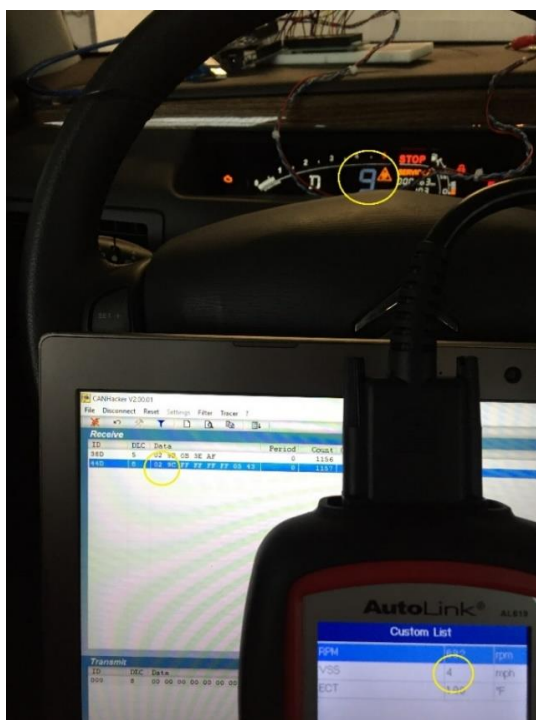


Рис.2.26. Реєстрація швидкості та перших двох байтів

**Швидкість на спідометрі та діагностичному пристрої**

Швидкість на сканері, тр/h	Конвертовано в km/h	Швидкість на панелі, km/h
16	25.75	28
19	30.58	33
28	44.8	48
32	51.5	55
34	54.72	58

Зафіксовано, декілька значень швидкостей та відповідні їм байти, для того, щоб розрахувати константу. Всі заміри здійснювались, як показано на (рис.2.26), тільки для різних значень швидкостей. Результати замірів наведено (таблиця 2.4). Надалі, будемо оперувати усіма значеннями швидкості, які відображались на діагностичному пристрої, але конвертованих у км/год.

Таблиця 2.4

**Значення кількості швидкостей та відповідні їм байти**

Швидкість, км/год	HEX формат	DEC формат	const.Швидкості	Середнє
6,437	0x02 та 0x9E	670	0,0096	0,0098
25,75	0x09 та 0xD0	2512	0,0102	
30,58	0x0C та 0x8E	3214	0,0095	
44,8	0x11 та 0xB2	4530	0,0098	
51,5	0x14 та 0x5F	5215	0,0098	
54,72	0x15 та 0x7A	5498	0,0099	

Наступним кроком, було переведено всі значення в десяткову систему числення та розраховано константу швидкості автомобіля Citroen C6 (табл. 2.4):

$$\text{const. швидкості} = \frac{\text{Значення швидкості}}{\text{DEC значення}} \quad (3.2)$$

В результаті було розраховано, середнє значення серед шести: 0,0098, таким чином, зараз є можливість розрахувати дійсну швидкість автомобіля.

Щоб розрахувати дійсну швидкість транспортного засобу потрібно перших два байти помножити на отриману константу (рис 2.27).

Серед великого масиву даних 1666 значень, побудовано графік (рис.2.28) та вибрано декілька проміжних точок, щоб навести приклад перетворення (табл. 2.5).

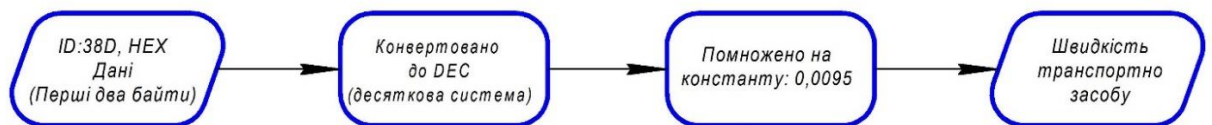


Рис.2.27. Алгоритм розрахунку швидкості транспортного засобу

Таблиця 2.5

### Розрахунок швидкості автомобіля

Час, мс	38D, повідомлення	Перших два байти	DEC формат	Дійсна швидкість,
3186	00495AF	00	0	0
12461	011D04B7B0	011D	285	2,7
22372	03CE051ABC	03CE	974	9,2
25078	0790058BAE	0790	1936	18,3
48921	0BF807EBC6	0BF8	3064	29,1

На (рис.2.28) відображено графік зміни швидкості автомобіля, під час їзди біля будівлі університету.



Рис.2.28. Графік зміни швидкості автомобіля

## **ВИСНОВКИ**

Проведено дослідження інтерфейсу CAN, що відповідає сучасним та актуальним вимогам студентської роботи у галузі автомобільна електроніка. На основі цього розроблено пристрій для аналізу інформації з шини обміну даних. Основні цілі даного аналізатора: отримувати, фільтрувати, записувати, частково відображати інформацію інтерфейсу.

Вивчено роботу протоколу CAN, виміряно осцилограми та проаналізовано природу сигналів шини комфорт та двигун. Застосовано заводські аналізатори Kvaser та Jonat, на основі них, визначено цілі для власного пристрою.

Знайдено методи ідентифікації та алгоритми розшифрування повідомлень для інтерпретування в звичний формат. Записано інформацію від різних сенсорів та блоків керування, яку в подальшому імпортовано в Excel та проаналізовано. Проведено декодування швидкості транспортного засобу та кількості обертів двигуна автомобіля Citroen C6.

Створено CAN аналізатор, основна відмінність якого - часткова автономність та можливість запису інформації.

Поставлено цілі, щодо вдосконалення пристрою - оновлення його до повної автономності та незалежності від комп'ютера.

Одна із можливих тем для реалізації в майбутньому, використати даний аналізатор для оцінки поведінки водія під впливом нормального водіння та відволікання. Для прикладу: порівнювати швидкість транспортного засобу при відволіканні і нейтральному стані на одній і тій самій дистанції. Також зчитувати в той самий момент часу інші характеристики автомобіля: оберти в хвилину, положення педалі газу, прискорення та інше.

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

- [1] Системы управления бензиновыми двигателями. Перевод с немецкого. С40 Первое русское издание. - М.: ООО "Книжное издательство "За рулем", 2005. - 432 с.: ил.
- [2] А. В. Заграничний, “Схемотехніка : Пристрої цифрової електроніки,” 2016.
- [3] K. Aminogatan and E. Ab, “Kvaser USBcan Rugged HS/HS Kvaser USBcan Rugged HS/HS,” pp. 7–10.
- [4] R. Baxter, N. Hastings, A. Law, and E. J. . Glass, “Arduino Uno R3 Datasheet,” *Datasheet Controll.*, 2008.
- [5] Microchip, “Mcp2515 Notes,” p. 94, 2003.
- [6] B. Earl, “Adafruit Data Logger Shield Installing the Headers Assembly with male headers Assembly with Stacking Headers : Prototyping Area Wiring & Config Older Shield Pinouts Older Datalogger Shield Leonardo & Mega Library Talking to the RTC First RTC test Setting ,” 2018.

## АНОТАЦІЯ

### **Актуальність теми.**

Автомобіль сьогодення є складним продуктом, який поєднує у собі різні галузі науки, такі як: механіка, електроніка та інформатика, які в подальшому створили нову галузь – мехатроніка. Предметом ускладнення конструкції транспортного засобу є вимоги щодо норм екології, безпеки, комфорту, діагностики та інше. У зв'язку з цим кількість застосованого електричного обладнання: блоків керування, сенсорів, виконавчих механізмів, довжина провідників зростає. Постійне збільшення електронного обладнання вимагає об'єднання у шини обміну даних – інтерфейси.

Відповідно до вдосконалення та модернізації транспортних засобів сьогодення виникла тема мого дослідження. Основні цілі якого дослідити шину обміну даних CAN та бути ближчим до інтерфейсів транспортних засобів, в науковому аспекті.

**Мета і задачі дослідження.** Метою дослідження є створення CAN аналізатора, який повинен повністю взаємодіяти з протоколом. На основі цього, виокремлено окремі задачі дослідження:

- виміряти осцилограми CAN комфорт/двигун, проаналізувати амплітуди та форми сигналів;
- застосувати заводські CAN аналізатори: Kvaser та Jonat, для розуміння роботи пристроїв;
- пошуки необхідного апаратного та програмного забезпечення для створення майбутнього, власного, аналізатора;
- створити перший прототип - надалі кінцевий продукт;
- отримати, відфільтрувати, відправити, записати повідомлення з CAN інтерфейсу;
- проаналізувати отримані дані.