НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і
спеціалізованих комп'ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ В.П. Тарасенко

«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050102 «Комп'ютерна інженерія»

на тему:

«Пристрій на базі GPS для вимірювання динаміки автомобіля.
Додаток для iOS»

Виконав (-ла):
студент (-ка) IV курсу, групи КВ-52
Побігай Антон Олександрович _____

Керівник:

_____

Консультант з нормоконтролю:

_____

Рецензент:

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050102 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ В.П. Тарасенко
«___»_____2019 р.

ЗАВДАННЯ

на дипломний проект студенту

Побігай Антон Олександрович

1. Тема проекту «Пристрій на базі GPS для вимірювання динаміки автомобіля. Додаток для iOS.», керівник проекту Коляда Костянтин В'ячеславович, асистент каф. СПСКС, затверджені наказом по університету від 22.05.2019 р. №1330-С

2. Термін подання студентом проекту «__» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз існуючих рішень, обґрунтування теми диплому;
   - особливості розробки пристрою на базі GPS для вимірювання динаміки автомобіля;
   - реалізація проекту;

5. Перелік обов'язкового графічного матеріалу:
   - Структурна схема;
   - Функціональна схема;
   - Схема роботи додатку;
   - Принципова схема пристрою.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль			

7. Дата видачі завдання «__» _____ 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	01.04.2019	
2.	Розроблення технічного завдання	05.04.2019	
3.	Розроблення структури проекту генератора	10.04.2019	
4.	Розроблення функціоналу обробки запитів на створення таблиць	14.04.2019	
5.	Підготовка матеріалів першого розділу дипломного проекту	24.04.2019	
6	Розробка апаратного забезпечення	30.04.2019	
7.	Підготовка матеріалів другого розділу дипломного проекту	05.05.2019	
8	Розробка програмного забезпечення для апаратного забезпечення	10.05.2019	
9.	Тестування роботи пристрою	15.05.2019	
10.	Підготовка матеріалів третього розділу дипломного проекту	20.05.2019	
11.	Розробка додатку для iOS	23.05.2019	
12.	Тестування додатку для iOS	25.05.2019	
13.	Підготовка графічної частини дипломного проекту	30.05.2019	
14.	Оформлення документації дипломного проекту	02.06.2019	

Студент А. О. Побігай

Керівник проекту К. В. Коляда

АНОТАЦІЯ

Об’єкт розробки – розробка власного пристрою та програмного забезпечення для організації передачі даних з нього. Створення програмного забезпечення для мобільної платформи для обробки отриманих даних з приладу.

Розроблений пристрій дозволяє:

- Отримувати швидкість за допомогою GPS модуля;
- Обчислювати прискорення автомобіля акселерометром;
- Бачити дані про зміну висоти за допомогою гіроскопу;
- Передавати дані за допомогою Bluetooth або Wi-Fi;
- Відключати GPS, акселерометр та гіроскоп, коли відсутню підключення до Bluetooth або Wi-Fi.

В пристрою передбачений захист від повного розряду батареї та обробка результатів на основі отриманих даних. В процесі розробки була використана мова програмування C, Swift з використанням бібліотеки SoftSerial та середовище розробки Arduino, Xcode. В якості серверу для обміну даними був розроблений особистий сервер.

В ході виконання дипломного проекту:

- розроблено апаратний засіб;
- проведено аналіз існуючих рішень;
- розроблено програмне забезпечення для підтримки роботи АЗ.

Створення такого пристрою націлено на подолання монополії на ринку любительських пристроїв для вимірювання швидкості та прибрати дефіцит цих приладів на ринку України.

Ключові слова: GPS, ESP32, C, SWIFT, ДИНАМІКА, ПРИСКОРЕННЯ, АКСЕЛЕРОМЕТР, ГІРОСКОП, ЧАСТОТА ОНОВЛЕННЯ, iOS, ARDUINO.

ABSTRACT

Object of project is development own hardware and software for transfer data by itself. Creating own software for mobile platform for using data from device.

Created device allow:

- Get speed using GPS;
- Counting of car acceleration;
- Look for changing in height using gyroscope;
- Send data using Bluetooth or Wi-Fi;
- When no connection with Bluetooth or Wi-Fi disable GPS, accelerometer.

Deep discharge protection and processing results using data. In development were used programming languages C/C++, Swift. SoftSerial library and Arduino IDE, Xcode were used.

In time of development:

- Created hardware;
- Analyzed of the finished solutions;
- Created software for working with hardware.

Creating of this device for the purpose to bit monopoly in market of amateurish device for checking of car performance. And to remove deficit of this devices in Ukraine.

Key words: GPS, ESP32, C/C++, SWIFT, PERFORMANCE METTER, ACCELERATION, ACCELEROMETER, GYROSKOPE, REFRESH RATE, iOS, ARDUINO.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ. 045490.002 ТЗ	Пристрій на базі GPS для	4		
			вимірювання динаміки			
			автомобіля.			
			Додаток для iOS			
			Технічне завдання			
	А4	ІАЛЦ. 045490.003 ТП	Пристрій на базі GPS для	2		
			вимірювання динаміки			
			автомобіля.			
			Додаток для iOS			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ. 045490.004 ПЗ	Пристрій на базі GPS для	59		
			вимірювання динаміки			
			автомобіля.			
			Додаток для iOS			
			Пояснювальна записка			
	А4	ІАЛЦ. 045490.005 Д1	Пристрій на базі GPS для	1		
			вимірювання динаміки			
			автомобіля.			

Зм	Лист	№ докум.	Підп	Дата	ІАЛЦ. 045490.001 ОА		
Розроб.		Побігай А.О.			Пристрій на базі GPS для вимірювання динаміки автомобіля. Додаток для iOS *Опис альбому*	Літ.	Лист 1 / Листів 2
Перев.		Коляда К.В.					
Н. контр.		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ, КВ-52	
Затв.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Додаток для iOS			
			Схема DRT.			
			Схема структурна			
	А4	ІАЛЦ. 045490.006 Д1	Пристрій на базі GPS для	1		
			вимірювання динаміки			
			автомобіля.			
			Додаток для iOS			
			Схеми алгоритмів роботи			
			Бездротової передачі даних			
			Схема структурна			
	А4	ІАЛЦ. 045490.007 Д1	Пристрій на базі GPS для	1		
			вимірювання динаміки			
			автомобіля.			
			Додаток для iOS			
			Алгоритм роботи			
			мобільного додатку			
			Для iOS			
			Схема структурна			
	А4	ІАЛЦ. 045490.008 Е3	Пристрій на базі GPS для	1		
			вимірювання динаміки			
			автомобіля.			
			Додаток для iOS			
			Схема взаємодії GPS та			
			мікроконтролера ESP32			
			Принципова схема			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Пристрій на базі GPS для вимірювання динаміки автомобіля. Додаток для iOS.».

Галузь застосування: обробка даних про динаміку та прискорення автомобіля.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення власного апаратного засобу для вимірювання динаміки автомобіля та програмний додаток для iOS.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- Можливість зберігати дані;
- можливість вибирати дистанцію для замірів;

- можливість вибрати швидкість замірів;
- аналітичні можливості.

5.2. Вимоги до апаратного забезпечення

- GPS модуль що працює на частоті в 19 Hz
- 3-х осьовий акселерометр та гіроскоп;
- Bluetooth/Wi-Fi модуль;

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система iOS старше 11 версії;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту
1.	Вивчення літератури за тематикою проекту	01.04.2019
2.	Розроблення технічного завдання	05.04.2019
3.	Розроблення структури проекту генератора	10.04.2019
4.	Розроблення функціоналу обробки запитів на створення таблиць	14.04.2019
5.	Підготовка матеріалів першого розділу дипломного проекту	24.04.2019
6	Розробка апаратного забезпечення	30.04.2019
7.	Підготовка матеріалів другого розділу дипломного проекту	05.05.2019
8	Розробка програмного забезпечення для апаратного забезпечення	10.05.2019
9.	Тестування роботи пристрою	15.05.2019
10.	Підготовка матеріалів третього розділу дипломного проекту	20.05.2019
11.	Розробка додатку для iOS	23.05.2019
12.	Тестування додатку для iOS	25.05.2019
13.	Підготовка графічної частини дипломного проекту	30.05.2019
14.	Оформлення документації дипломного проекту	02.06.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ. 045490.004 ПЗ	Пристрій на базі GPS для	59		
			вимірювання динаміки			
			автомобіля.			
			Додаток для iOS			
			Пояснювальна записка			
	А4	ІАЛЦ. 045490.005 Д1	Пристрій на базі GPS для	1		
			вимірювання динаміки			
			автомобіля.			
			Додаток для iOS			
			Схема DRT.			
			Схема структурна			
	А4	ІАЛЦ. 045490.006 Д1	Пристрій на базі GPS для	1		
			вимірювання динаміки			
			автомобіля.			
			Схеми алгоритмів роботи			
			Бездротової передачі даних			
			Схема структурна			
	А4	ІАЛЦ. 045490.007 Д1	Пристрій на базі GPS для	1		
			вимірювання динаміки			

					ІАЛЦ. 045490.003 ТП	
Зм	***Лист***	***№ докум.***	***Підп***	***Дата***		
Розроб.		Побігай А.О.			Пристрій на базі GPS для вимірювання динаміки автомобіля. Додаток для iOS ***Відомість технічного проекту.***	***Літ.*** ***Лист*** ***Листів*** 1 2
Перев.		Коляда К.В.				
						КПІ ім. Ігоря Сікорського, ФПМ, КВ-52
Н. контр.		Клятченко Я.М.				
Затв.		Тарасенко В.П.				

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Додаток для iOS			
			Алгоритм роботи			
			мобільного додатку			
			Для iOS			
			Схема структурна			
	А4	ІАЛЦ. 045490.008 Е3	Пристрій на базі GPS для	1		
			вимірювання динаміки			
			автомобіля.			
			Додаток для iOS			
			Схема взаємодії GPS та			
			мікроконтролера ESP32			
			Принципова схема			
			Додаток для iOS			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ. 045490.003 ТП	Арк. 2

ЗМІСТ

стор.

ДОДАТКИ

Додаток 1. Копії графічних матеріалів

ІАЛЦ.045490.005 Д1. Схема DRT. Схема структурна.

ІАЛЦ. 045490.006 Д1. Схема алгоритмів роботи бездротової передачі даних. Схема структурна.

ІАЛЦ.045490.007 Д1. Алгоритм роботи мобільного додатку для iOS. Структурна схема.

ІАЛЦ. 045490.008 Е3. Схема вазємодії GPS та мікроконтролера ESP32. Принципова схема.

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

АЗ – апаратне забезпечення

ПЗ – програмне забезпечення

BLE (Bluetooth Low Energy) – блютуз з низьким енергоспоживанням

DRT (Drag Racing Tool) – засіб для драг рейсінгу

GPS (global positioning system) – глобальна система позиціонування

MVC (Model-View-Controller) – Модель-Інтерфейс-Контролер

SPI (Serial Peripheral Interface) – послідовний периферійний інтерфейс

I2C (Inter-Integrated Circuit) – міжінтегральна схема

ВСТУП

Стрімкий розвиток мобільних технологій дає змогу розробникам створювати власні програмні та апаратні засоби для вимірювання швидкісних характеристик автомобіля. Ці засоби називають вимірювачами потужності (performance meter). Вимірювач потужності - це апаратний та програмний засіб, що характеризує динамічні можливості автомобіля. Ці пристрої дають змогу визначати час прискорення, проходження дистанцій та перевантаження під час прискорення.

Метою цих вимірювань є отримання об’єктивних значень, аналізуючи яких можна оцінити співвідношення маси до потужності, ефективність реалізації крутного моменту в залежності від трансмісії автомобіля, дорожніх умов, погодних показників та гуми.

Серед найпопулярніших пристроїв є Racelogic, Dragon, Dragy GPS Performance Meter та інші.

Також для якісного порівняння характеристик автомобіля використовується такі засоби, як акселерометр, гіроскоп та термометр.

Необхідність в ПЗ для конкретного пристрою прямо пов’язана від апаратного забезпечення, що використовується для його проектування. Відповідно, кожний пристрій має своє програмне забезпечення та результати вимірювань можуть відрізнятися одне від одного.

В даному дипломному проекті описано ПЗ, що обробляє дані отримані від пристрою по протокол Bluetooth 4.0 BLE та АЗ, що надає ці дані.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1 Загальний опис проблеми

Проблема що полягає в даному дипломному проекті - розробка дешевого пристрою для вимірювання динаміки автомобіля.

Через монополію, що тривала впродовж 17 років на ринку приладів для моторспорту досить просто знайти потрібний апаратний засіб, але дуже складно знайти в нашій країні. Для покупки необхідно гаджета було необхідна замовляти його з іншої країни, оплачувати доставку та розмитнення. Це створює значні труднощі для покупця, оскільки за доставку та розмитнення треба заплатити ще близько 30% від початкової вартості та чекати впродовж 3-5 тижнів на доставку пристрою до покупця. Купуючи засіб вже в Україні продавець вже враховує витрати на доставку та розмитнення та закладає в ціну свій прибуток. Але даний пристрій навіть не мав програми для телефону і не оновлювався з 2002 року та дуже громіздкий. В користувача навіть нема можливості повторно подивитися результат заїзду і живлення не було автономним.

Останні 5 років дану нішу пробували зайняти різні виробники зі своїми рішеннями, які були значно компактніші, дешевші, але залишалися дуже не зручними та досить дорогими. Все ще не було можливості повторно дивитися результати, була відсутня програма для телефону та живлення все ще відбувалося від машини або зовнішнього зарядного пристрою.

2 роки тому китайським розробникам вдалося створити конкурента та усунути монополію з ринку пристроїв для вимірювання динаміки автомобілів. Новий продукт вже мав дуже зручну програму для iOS та Android, мав вбудовану батарею, що дозволяло пристрою існувати автономно впродовж 7 годин роботи, компактний корпус. Більше не потрібно було фіксувати прилад до скла і заважати водію під час заїзду. Роздрібна ціна менша в 4 рази. Але у

пристрою залишилися недоліки - практична відсутність в магазинах, тому що вартість розмитнення у відсотковому співвідношенні більша. Через штучно створений дефіцит товару продавці завищують вартість засобу, а покупці відразу купують.

Досить важливим фактором все ще залишається сприйняття людьми цих пристроїв, тому що за еталонні значення всі беруть показники пристрою розробленого в далекому 2001 першому році. Не зважаючи уваги на те, що сучасні пристрою мають точніші вимірювання за рахунок частішого оновлення даних та використання сучасних технологій при розробці.

Отже, можна виділити наступні проблеми:

-висока ціна;

-відсутність пропозиції на полицях магазинів;

-довготривала доставка;

-дороге розмитнення;

-низька конкурентність на ринку;

-скупий функціонал програмних рішень;

-не можливість оновлення програмного забезпечення для апаратного засобу.

1.2 Аналіз існуючих рішень

Для аналізу існуючих рішень слід виділити 4 прилади: Racelogic VBOX Sport, Racelogic PerformanceBox, DragOn, Dragy.

Існує безліч інших пристроїв. В цілому вони повторюють функціонал DragOn. Серійне виробництво цих пристроїв не існує.

Слід виділити пристрій на базі ELM327, що являє собою OBD конектор і отримує дані про швидкість з автомобіля. Оскільки дані про швидкість можуть різнитися в залежності від розміру коліс автомобіля та передатних чисел коробки переключення передач.

	vBox Sport	PerformanceBox	Dragy	DragOn
				
Частота оновлень	20 Hz	10 Hz	10 Hz	10 Hz
Вбудована батарея	Так	Ні	Так	Ні
Вбудована GPS антена	Так	Так	Так	Так
Накопичувач пам’яті	SD карта	SD карта	Внутрішня пам’ять	SD карта
Обмін даними через Bluetooth	Так	Ні	Так	Ні
Захист від попадання вологи	Так	Ні	Ні	Ні
Додаток для телефону	Так	Ні	Так	Ні
Розмір	104x72x25 мм	113x63x93 мм	76x25x14 мм	95x33x22 мм
Вага	130 грам	225 грам	105 грам	120 грам
Ціна	450$	500$	150$	200$

	vBox Sport	PerformanceBox	Dragy	DragOn
				
Матеріали корпусу	Полікарбонат	АБС пластик	Полікарбонат	АБС пластик
Дисплей	Ні	Так	Ні	Так
Виміри часу кола	Так	Так	Ні	Ні
Додаткова антена	Так	Так	Ні	Ні

VBOX Sport

Технічні специфікації

Швидкість:

Точність: 0,1 км/год

Одиниці вимірювання: км/год, мл/год

Частота оновлень: 20 Hz

Мінімальна швидкість: 0,1 км/год

Максимальна швидкість: 1600 км/год

Прискорення:

Точність: 5%

Максимум: 4 G

Навколишнє середовище та фізичні якості:

Вхідне живлення: лише +5 V

Потужність: 2,5 Wat максимум (під час зарядки)

Рівень захисту: IP65 (з закритими портами підключень), IP20 (з відкритими портами підключень)

Розмір: 104 mm x 72 mm x 25 mm

Вага: 130 грам

Умови зберігання: від -20º C до +60º C

Програмне забезпечення:

Сумісна робота з: iPhone, iPad, Windows

PerformanceBox

Технічні специфікації

Швидкість:

Точність: 0,1 км/год (усереднено по 4 зразкам)

Одиниці вимірювання: км/год або мл/год

Частота оновлень: 10 Hz

Мінімальна швидкість: 0,1 км/год

Максимальна швидкість: 1600 км/год

Затримка: < 160 мс

Дистанція:

Точність: 0,05% (<50см на кілометр)

Одиниці вимірювання: Метри/фути

Прискорення:

Точність: 1%

Максимум: 4 G

Навколишнє середовище та фізичні якості:

Вхідне живлення: лише 6-28 V постійного струму

Потужність: 100 mA, +12 V підключено до автомобіля

Розмір: 113 mm x 63 mm x 93 mm

Вага: 225 грам

Умови зберігання: від -30º C до +80º C

Dragy

Технічні специфікації

Швидкість:

Точність: 0,01 км/год

Одиниці вимірювання: км/год, мл/год

Частота оновлення: 10 Hz

Мінімальна швидкість: 0,01 км/год

Прискорення:

Точність: 1%

Максимум: 5G

Навколишнє середовище та фізичні властивості:

Вхідне живлення: +3,7 V

Розмір: 76 mm x 25 mm x 14 mm

Вага: 105 грам

Умови зберігання: від -20º C до +60º C

Програмне забезпечення:

Сумісна робота з: Android 4.4 та вище, iOS 8 та вище

DragOn

Технічні специфікації

Швидкість:

Точність: 0,1 км/год

Одиниці вимірювання: км/год

Частота оновлення: 10 Hz

Мінімальна швидкість: 0,1 км/год

Навколишнє середовище та фізичні властивості:

Вхідне живлення +5 V

Розмір: 95 mm x 33 mm x 22 mm

Вага: 120 грам

Умови зберігання: від -20º С до +60º С

Інформація взята з сайту: drive2.ru

Варто зазначити, що VBOX Sport, PerformanceBox, DragOn не мають можливості проводити заміри 100-200 км/год без повної зупинки автомобіля, коли dragy не маючи вбудованого процесора для обробки даних так сказати «на борту» передає всі показники на телефон і всі обчислення відбуваються на телефоні. За допомогою цього у розробників з’являється змога отримання більш точних вимірювань, можливість навантажувати потужний телефон, а не слабкий прилад та заощаджувати використання батареї, що дозволяє користувачу довше використовувати свій пристрій.

Необхідно зауважити, що до динаміки автомобіля враховується не лише динаміка розгону автомобіля, а і гальмівну динаміку. Всі засоби, що були вище названі, також оснащені функцією вимірювання гальмівної динаміки. Вище це

не було описано, тому що ці вимірювання не є об’єктивними. Гальмівна динаміка заміряє час гальмування зі 100 до 0 км/год. Ці вимірювання не слід брати до уваги, тому що користувач може заміряти динаміку автомобіля на четверть милі, а потім продовжувати їхати в спокійному режимі, а зупинитися вже на світлофорі. Для коректного вимірювання даного типу динаміки треба використовувати акселерометр, як і для прискорення. За допомогою акселерометра прилад починає відлік часу про розгін і дає можливість побачити яке перевантаження відчуває користувач в момент старту. За тією ж логікою слід обчислювати дані про гальмування. Автомобіль повинен їхати зі швидкістю 100 км/год за даними пристрою, а не за показниками спідометру, потім водію необхідно натиснути на гальма до повної зупинки машини. Оскільки така зупинка може спровокувати дорожньо-транспортну пригоду, не рекомендовано проводити такий тест.

Також слід виділити програмні забезпечення, що дозволяють проводити заміри динаміки автомобіля. Ці додатки отримують дані про швидкість з GPS модуля вбудованого в телефон. Оскільки операційна система телефону працюю з перериваннями, то дані про швидкість приходять зі значною затримкою і має дуже велику частоту оновлень, тому що операційна система може відключати GPS модуль. Через це дані про заїзд будуть не об’єктивними, тому що в результати заміру не враховуються можливі затримки через переривання, що виникають.

В порівнянні з цими програмними забезпеченнями можна розглянути OBD пристрій, що базується на чіпі ELM327. Тому що він хоча б працює без переривань, але затримка в передачі даних через Bluetooth досить велика, а час засікає телефон. Таким вимірюванням також не можна довіряти, тому що дані про швидкість пристрій бере з датчика ABS, він може мати похибку в 5 км/год. В професійній та любительській сфері цей прилад не використовується через його помилку у вирахуванні результатів.

1.3 Постановка задачі

Головна ціль дипломного проекту - створити конкурентно спроможний пристрій для вимірювання динаміки автомобіля.

Проаналізувавши всі доступні продукти на ринку, врахувавши всі їхні переваги та недоліки, були виділені основні апаратні та програмні вимоги для реалізації свого пристрою.

Ключові вимоги до апаратного та програмного засобу:

-висока швидкість оновлення даних з GPS-пристрою;

-зручна програма для телефону(планшету) користувача;

-компактний корпус для апаратного засобу;

-довготривала автономна робота пристрою від вбудованої акумуляторної батареї;

-доступна ціна;

-захист від попадання пилу та вологи.

Задача полягає в проектуванні апаратного забезпечення, розробки програмного забезпечення, що буде керувати апаратним засобом. Розробка додатку для мобільної платформи.

Необхідність розробки особистого апаратного засобу полягає в бажанні створити прилад, який зможе конкурувати з іншими пристроями і в першу чергу показати якісно точніші показники ніж конкуренти. Власна платформа дозволить розробнику оптимізувати програмне забезпечення під апаратне та цим якісно конкурувати з діючими лідерами даної ніші на ринку засобів для моторспорту.

2. ОСОБЛИВОСТІ РОЗРОБКИ ПРИСТРОЮ НА БАЗІ GPS ДЛЯ ВИМІРЮВАННЯ ДИНАМІКИ АВТОМОБІЛЯ

2.1 Аналіз особливостей даної розробки

Для зручності висловлювань та подальшого комерційного використання дамо назву проекту – DRT (drag racing tool).

В цьому проекті було використано досить багато унікальних рішень, які раніше не були ніким застосовані. До таких рішень можна віднести:

-використання плати ESP32 з вбудованим Bluetooth/Wi-Fi модулем. Особливість цієї плати полягає в тому що окрім головного ядра в платі, Bluetooth модуль має окреме ядро, що дозволить синхронізувати роботу. Оскільки основне ядро є потужнішим, є сенс його задіяти для обробки даних, що надходитимуть з GPS модуля. Допоміжне ядро не зважаючи на низьку продуктивність має достатньо потужності для оброки даних, що надходитимуть з акселерометра та гіроскопа. Акселерометр та гіроскоп має один корпус та дані з них передаються одночасно з великою частотою;

-робота GPS на частоті 19 Hz для точних вимірювань. При налаштуваннях були отримано максимальне значення, що становило 19 Hz. Результат можна назвати задовільним, оскільки це краще ніж у більшості конкурентів;

-можливість оновлення програмного оновлення для апаратного забезпечення, для усунення помилок, що виникли після кінцевого тестування пристрою;

-програмне відключення GPS та акселерометра для економії батареї;

-програмне забезпечення розроблене окремо для iOS та Android, що дозволить якісно оптимізувати роботу для обох операційних систем. Оскільки кожна операційна система має свої особливості, то окрема розробка дозволить покращити швидкодію та безпеку програми.

2.2. Архітектурні аспекти DRT

При розробці цього проекту було розроблено та випробувано 3 різні конфігурації пристрою.

1) Пристрій складався з таких компонентів:
   - процесор Atmega 328p;
   - Bluetooth HC-05;
   - GPS Ublox NEO-6M-0-001;
   - Гіроскоп, акселерометр, термометр;

   До переваг можна віднести:

   -програмування мікроконтролера;

   -швидка заміна не працюючих елементів плати;

   -низька вартість комплектуючих;

   -існує багато корпусів для процесора.

   Недоліки:

   -мікроконтролер можна легко пошкодити;

   -HC-05 не сумісний з iOS пристроями;

   -процесор маж лише 1 потік;

   -часто виникає нестача струму на окремих модулях.

При розробці виникало досить багато складнощів, що не давало проекту зрушитися з місця. Середовище розробки вимагало спеціальне апаратне забезпечення для запису інформації на мікроконтроллер. Модулям зазвичай не вистачало вхідного живлення для стабільної та безвідмовної роботи модулів Bluetooth та GPS. Оскільки існує багато корпусів для Atmega 328P, то в фінальній реалізації пристрою можна буде значно зекономити місця на монтажній платі.

Прорахувати на перед собівартість такого пристрою стало не реально, оскільки пристрій складається з безлічі транзисторів, діодів, резисторів і так далі. Такі дрібні деталі можна порахувати вже після успішного закінчення проекту. Такий варіант проекту був найдорожчим по собівартості та

найскладнішим в реалізації за решту, оскільки вимагав від розробника глибоких знань в мікроелектроніці. Дана конфігурація має своє право на життя, оскільки звільняє розробника від залежності від сторонніх бібліотек, які є обов'язковими для підключення того чи іншого модулю до плати.

HC-05 являє собою базовий модуль з антеною. Головним недоліком стало те, що він працює за протоколом Bluetooth 2, 3, але розробка АЗ для iOS вимагає Bluetooth 4.0 BLE. Це можна вирішити лише заміною Bluetooth модуля. Також слід зауважити, що можна було Bluetooth замінити на Wi-Fi, що дозволило б підключення обох платформ без додаткових труднощів і швидкість оновлення даних була б значно вищою. Але в даному проекті, з інженерної точки зору та з точки зору зручності в користуванні було доцільніше використовувати Bluetooth, оскільки підключитися до нього можна не згортаючи програму, на відміну від Wi-Fi [11].

Найкращим рішенням всіх труднощів, що виникли в процесорі роботи стало – заміна процесора на плату типу Arduino, Raspberry Pi та інші, що і було зроблено. Таке рішення звільнило від розробки всієї схеми, підбору транзисторів, діодів, резисторів і так далі.

Безумовно цю конфігурацію можна назвати найцікавішою з точки зору розробки, але після згорання одного процесору та блокування другого стало зрозуміло, що цей варіант займе надзвичайно багато часу на розробку та реалізацію.

2) Пристрій під управлінням Arduino:
   - Репліка Arduino Nano;
   - Bluetooth BT-05;
   - GPS Ublox NEO-6M-0-001;
   - Гіроскоп, акселерометр, термометр;
   - Зарядний пристрій з micro-USB;
   - Акумуляторна батарея.

   Переваги:

- Зручне середовище розробки програмного забезпечення для Arduino;
- Вільний доступ до бібліотек для підключення модулів;
- Для програмування модулів використовується Micro USB;
- Малі розміри самої плани Arduino Nano;
- Захист від перенапруги;
- Всі модулі на вході мають однакову напругу.

Недоліки:

- Одноядерний процесор;
- Не можлива реалізація одночасної паралельної роботи 2 та більше модулів;
- Залежність від обов'язкових бібліотек для передачі даних від модулю до Arduino.

Було використано репліку Arduino Nano, а не оригінал, тому що вони різняться лише ціною та кількістю заводського браку, тож для цілей проекту його було більш ніж достатньо.

За використанням цієї конфігурації було отримано багато позитивних результатів, що проявило світло на велику кількість речей, що раніше були не відомі та не доступні для використання. Вперше стало зрозуміло, що стандартна швидкість оновлення GPS модуля, що використовувався, замала для точних вимірювань. При використанні акселерометра обов'язково отримувати показники з осі Z. Без використання цієї осі не було змоги записати програмний код до плати.

Були визначені оптимальні апаратні рішення для даного пристрою, що дозволило зняти багато питань, що стосувалися розробки і дозволило рухатися далі. Вже на цьому етапі стало зрозуміло, наскільки використання сторонніх блат економить час та ресурси розробника для створення продукту та своїх особливостей, якщо час, як в даному випадку, є обмеженим.

З'явилась можливість сформувати весь перелік вимог до проекту та зрозуміти можливості для їх реалізації. Однією з таких вимог стала організація

паралельної роботи GPS та акселерометра з гіроскопом. Без організації переривань реалізація цього стала не можливою, відповідно знизилась цінність проекту, як комерційного продукту.

Налаштувавши модуль GPS за допомогою програмного забезпечення, що надає виробник цього модулю, було отримане значення в 10 Hz та стало зрозуміло, що можна отримати швидкість оновлення, що буде рівна 19 Hz. Дана швидкість не є більшою ніж у VBox Sport, але є більшою ніж у більшості конкурентів. Дана швидкість є більш ніж достатньою для такого типу пристрою, оскільки цей проект є швидше для звичайного користувача ніж для спортсменів.

При змозі налаштування нашого модулю для роботи з Arduino стало зрозуміло, що бібліотека, що використовувалась для передачі даних між платою та модулем не здатна працювати на такій частоті. Це стало ключовим питання для зміни плати Arduino на іншу, оскільки в такому разі цінність та доцільність розробки втрачається. Використання інших бібліотек не є можливим, тому що вони є старішими і не підходили під версію програмного забезпечення Arduino [14].

3) Пристрій під управлінням плати на основі Bluetooth ESP32:
   - Плата з 2-во ядровим Bluetooth ESP32;
   - GPS Ublox NEO-6M-0-001;
   - Гіроскоп, акселерометр, термометр;
   - Зарядний пристрій з micro-USB;
   - Акумуляторна батарея.

   Переваги:
   - 2-во ядровий процесор;
   - можливість організації паралельної роботи 2 модулів;
   - собівартість апаратного забезпечення значно зменшилась;
   - одночасна робота Bluetooth та Wi-Fi.

Недоліки:

- Більший розмір плати;
- Вищі температурні режими роботи;
- Складно купити в Україні, доставка плати триватиме 2-3 тижні.

За допомогою 2-во ядрового процесора нарешті з’явилась змога одночасно отримувати дані з 2 незалежних модулів, що було необхідно з самого початку роботи над проектом. Оновлення програмного забезпечення стало можливо передавати через Wi-Fi. За допомогою цього стало можливо після змін в програмному забезпеченні проводити перепрограмування плати віддалено, розмістивши плату з модулями біля вікна, оскільки GPS модуль погано отримує зв’язок із супутниками в будівлях з панельним або залізобетонним покриттям. З використанням Arduino було необхідно весь час бути біля вікна, щоб модуль мав зв’язок або після кожних змін ходити та ловити зв’язок по 3-5 хвилин.

Головною перевагою ESP32 є можливість відключати живлення від модулів, якщо відсутнє Bluetooth або Wi-Fi підключення, програмно з використанням транзисторів, що дає змогу заощадити роботу акумуляторної батареї. Оскільки ціна на ESP32 є така ж сама, як і репліки Arduino Nano, але ESP це і є Bluetooth/Wi-Fi модуль, то собівартість пристрою вдалось істотно зменшити, що позитивно вплинуло на комерційну складову даного проекту. Але більші розміри плати змушують переглянути розміщення всіх елементів на монтажній платі, оскільки розпайка на текстолітовій пластині відбувається вручну паяльником.

Робота з акселерометром, гіроскопом та GPS стала більш незалежною одне від одного, шляхом розділення їх на різні потоки. Повної незалежності не вдалося досягти, оскільки Bluetooth вимушений одночасно приймати та передавати дані з 2 різних модулів.

Для тестування різних бібліотек для роботи з GPS був куплений додатковий. Один працював з частотою 1 Hz, а другий з частотою 19 Hz. Це

дозволило організувати «гарячу» заміну модулів, змінивши порт підключення програмним забезпеченням, що є дуже зручно в процесі розробки. Оскільки підбір бібліотек для взаємодії керуючого пристрою ESP та GPS модулю стало куди зручнішим та швидким процесом.

Головною проблемою даної конфігурації стали високі температурні режими роботи плати, що спонукало до організації достатньої терморегуляції корпусу, що змусило відмовитися на певний час від пило-волого захисту. Переконаний, що це питання буде вирішено в пізніших ревізіях пристрою, оскільки в реальних умовах прилад буде використовуватися в сніг, дощ, мороз, спеку і так далі.

Все одно ця конфігурація є найбільш вдалою на даному етапі за 2 попередні. Можливо в майбутньому доведеться відмовитися від даної плати на більш потужну, використати інший GPS модуль. Але станом на сьогодні не має в цьому потреби. Більш потужні плати є значно дорожчими та громіздкими, що робить пристрій менш зручнішим та привабливим для покупки.

2.3 Проектування та організація структури DRT.

В основу всього проекту було покладено GPS модуль Ublox NEO-6M, що є пристроєм початкового класу, але достатнім для вимог даного проекту. Модуль стандартно працює на частоті в 1 Hz. Компанія Ublox надає програмне забезпечення за допомогою якого можна здійснити налаштування пристрою на необхідну для використання швидкість. Щоб налаштувати прилад необхідна підключити GPS до комп’ютера за допомогою USB-UART адаптера. Не зважаючи на те що даний модуль не є професіональним, він здатний працювати на частоті в 19 Hz. Така частота є не найшвидшою у порівнянні з вже існуючими рішеннями. Ця частота є краще на 90% за більшість конкурентів та уступає лише одному приладу. GPS модуль оснащений антеною в керамічному корпусі з можливістю від’єднання від плати. На самій

схемі та антені передбачені місця для розпайки антени на платі. На корпусі пристрою передбачені технологічні отвори для проведення проводів або для кріплення схеми до корпусу пристрою болтами. На платі присутній світловий діод для індикації зв'язку з супутниками. Якщо з'єднання з супутником не встановлено – індикатор не працюватиме.

Керуючим пристроєм є плата ESP32 з вбудованим Wi-Fi/Bluetooth модулем. Схема має 18 цифрових контактів для зчитування та запису інформації на зовнішні модулі. Запис даних для управління всією системою відбувається через Micro-USB роз'єм. Плата має вбудований модуль Wi-Fi та Bluetooth зі своєю антеною, місце для підключення зовнішньої антени не передбачене. При проектуванні була вибрана саме ця плата в якості керуючого пристрою, оскільки розробка програмного забезпечення відбувається за допомогою середовища розробки Arduino та апаратно оснащений 2-во ядерним процесором, що дозволяю легко та зручно організувати паралельну роботу зовнішніх модулів не прибігаючи до переривань [3]. На платі є 2 контакти для землі, що з'єднанні між собою. Передбачений контакт для зовнішнього живлення. Недоліком плати є її великі розміри та високі температурні режими роботи Wi-Fi/Bluetooth модуля. На корпусі пристрою передбачені технологічні отвори для проведення проводів або для кріплення схеми до корпусу пристрою болтами. Завдяки використанню цього пристрою з'являється можливість відключати живлення на модулі, якщо відсутнє з'єднання з користувачем. Це дозволяє економити заряд акумуляторної батареї та цим продовжує час автономної роботи всього приладу, що є дуже зручно з точки зору використання. Але повне відключення живлення призводить до переходу GPS модулю в холодний режим, що негативно відображається на часі запуску цього пристрою. Для вирішення цієї проблеми було прийнято рішення задіяти біполярний транзистор npn типу та резистор на 10 КОм. Коли є підключення по Bluetooth струм йде через транзистор на GPS та акселерометр з гіроскопом тим самим забезпечує високий рівень струму

необхідний для роботи пристрою. Коли ж підключення відсутнє транзистор починає працювати в іншому режимі та подає низький рівень струму на модулі, це не відключає їх, але і не споживає багато напруги акумуляторної батареї, тим самим забезпечує довготривалу автономну роботу всього пристрою.

Для вимірювання прискорення та зміни висоти використовується 3-х осьовий акселерометр та 3-х осьовий гіроскоп, який оснований на базі плати MPU6050. Для якісного аналізу вимірювань слід мати інформацію про зміну висоти, оскільки це може бути ключовим фактором у вимірюваннях. Акселерометр необхідний для того, щоб вимірювати навантаження під час прискорення та впродовж всього заїзду. Дані про навантаження не завжди є необхідними для користувача. Необхідність в цих даних виникає після змін в потужності автомобіля. Це дає змогу аналізувати про зміни в динаміці автомобіля. Модуль також оснащений термометром, але необхідності в його використанні нема, тому що він буде відображати температуру в середині пристрою, що не є потрібним для користувача, але на етапі проектування є необхідним для розміщення модулів в корпусі. Його використання дозволяє аналізувати яка температура в середині пристрою і розміщати модулі так, щоб прилади не перегрівалися та не виходили з ладу. Оскільки гіроскоп та акселерометр є 3-х осьовими, то при розробці можна визначати як буде розміщено модуль в платі, горизонтально або вертикально та програмно це зазначити. На модулі є 8 контактів серед яких лінія даних I2C та лінія синхроімпульсів I2C. Також передбачений контакт для налаштованих переривань та 2 додаткових I2C інтерфейси для підключення зовнішнього магнітометра.

Для організації автономної роботи пристрою використовується акумуляторна батарея Li-Po типу на 600 mAh на 3,7 вольт. Був вибраний саме цей тип батареї, оскільки такі акумуляторні батареї в менший розмір вміщають більший об’єм заряду. В якості зарядного пристрою та для контролю від

глибокого розряду використовується зарядний пристрій з захистом. В прилад вбудований micro-USB роз'єм для заряджання. Використовується саме такий формат входу, тому що він є найбільш поширеним і терміново знайти провід для зарядки не стане проблемою. Можливо заряджати пристрій через передбачені для цього контакти, але в цьому не виникає необхідності, оскільки передбачено сучасний роз'єм для цього. На платі передбачені контакти для з'єднання з керуючим пристроєм для передачі живлення та про стан заряду батареї. На схемі є 2 світлових діода, які працюють лише під час заряджання батареї і сповіщають про повний заряд. Прилад має досить компактний роз'єм і на монтажній платі може бути розміщений під керуючий пристрій. Недоліком плати є високий температурний режим під час заряджання.

Після всіх тестувань та налаштувань всі модулі буде розміщено та вручну розпаяні на текстолітовій пластині з отворами покритими міддю. Кінцеву плату буде поміщено в міцний та герметичний корпус для того щоб захистити від падінь та попаданню вологи в середину, оскільки це є необхідною мірою безпеки при використанні будь якого пристрою.

2.4 Технологія пристрою.

Під час проектування було задіяно досить багато технологій, що позитивно вплинуло на прилад. Це були як програмні, так і апаратні рішення. Впровадження сучасних технологій є надзвичайно важливим фактором для створення конкуренто спроможного комерційного пристрою, що дасть можливість отримати сучасні функції та максимально спростить роботу з приладом та додатком. Використання технологій в апаратних рішеннях вплинуло на собівартість товару, на швидкість роботи пристрою та якісно покращила результатів. Технології в програмній галузі були націлені на безпеку, швидкодію та коректну оброку отриманих результатів. Головна складність виникла в тому, що певні рішення перешкоджали роботі інших. Для вирішення таких питань проект доводилося навіть змінювати концептуально.

Технологією з точки зору апаратного рішення можна назвати:

-використання керуючої плати з 2-во ядерним процесором для запобігання переривань;

При реалізації дипломного проекту було прийняте рішення використовувати сучасну плату керування в основу якої лягає Bluetooth/Wi-Fi модуль. Це дозволяє економити місце в корпусі пристрою та ергономічно розмістити всі модулі на монтажній платі з текстоліту. Технологічна особливість цієї плати полягає в тому, що ця плата оснащена 2-ма ядрами. Це надає можливість організувати синхронну роботу всіх модулів, що використовуються на платі. Не прибігаючи до переривань у випадку роботи з іншими керуючими пристроями. Впровадження керуючого пристрою з вбудованим Bluetooth/Wi-Fi модулем також дає змогу суттєво скоротити собівартість пристрою на етапі розробки та подальшого серійного випуску.

-використання 3-ох осьових акселерометра та гіроскопа для зручного розміщення в корпусі пристрою;

Цей модуль націлений на вимірювання перевантажень під час розгону автомобіля та визначення змін висоти під час заїзду. Сигнал про відлік часу дає саме акселерометр, оскільки GPS може мати похибку. Як тільки в отриманих даних з акселерометра значення перевантаження змінюється на більше, то починається час відліку. Це дає змогу прибрати похибку супутників та мати якісно кращий результат. Оскільки акселерометр є 3-ох осьовим, то він може бути розміщений як вертикально, так і горизонтально. Все що буде необхідно змінити – вісь координат в самій програмі. Гіроскоп необхідний для вимірювання висоти, оскільки під час заїзду це є важливим фактором. Кожна зміна висоти впливає на кінцевий результат.

-використання Li-Po батареї, це дозволило зекономити місце та отримати більшу ємність батареї;

З метою економії місця в корпусі пристрою було вирішено використати акумуляторну батарею типу Li-Po. Перевагою цього типу батареї є те що вона

вміщує більшу кількість заряду в меншу площу ніж Li-Ion батареї. В корпус розміром 40x25x5 mm вміщається 620 mAh. Це дозволяє мати більшу автономну роботу ніж у конкурентів при менших розмірах пристрою. Головним недоліком акумуляторної батареї такого типу є те, що вони погано реагують на низький заряд, що може призвести до нестабільної роботи у майбутньому. Для цього довелось використати наступну технологію.

-захист від перенапруги та критично низького заряду шляхом використання додаткових захисних модулів;

Контролер заряду є обов'язковим модулем для вдалої реалізації такої конфігурації, оскільки Li-Po батарея досить вибаглива до рівня заряду, тому було необхідно використати цей модуль. Він не дозволяє батареї перейти в «глибокий» розряд цим забезпечує її подальшу безвідмовну роботу. Цей модуль також захищає плату від перенапруги, що є беззаперечно позитивною властивістю цього модулю. Найголовнішою перевагою модуля є те, що він також є зарядним пристроєм для всього приладу.

-відключення модулів GPS, акселерометра, гіроскопа тоді, коли нема з'єднання з гаджетом користувача з метою економії заряду акумуляторної батареї;

Можливість зменшувати живлення, що подається на модулі є надзвичайно необхідною функцією. Цим можна досягти довготривалу роботу пристрою та мінімізувати час запуску GPS. Це досягається шляхом встановлення додаткових транзисторів в схему. Під час відключення з'єднання з телефоном або планшетом користувача, транзистори переходять в закритий режим і на модулі поступає низький рівень струму. У випадку роботи з акселерометром та гіроскопом навіть повне відключення живлення не є проблемою, оскільки після відновлення живлення вони відразу запускаються. Нажаль, GPS має властивість переходити в режим «холодної» роботи і запуск його з цього стану може тривати дуже багато часу, що не є зручно та не відповідає вимогам швидкодії апаратного забезпечення.

-використання сучасного micro-USB порту для обміну даних на етапі розробки та для зарядження пристрою;

У сучасному світі дуже важливим фактором є роз'єм для живлення, оскільки майже кожна людина має з собою 1-2 гаджета та павербанк для зарядки пристрою та провід для зарядки саме свого телефона. Слід врахувати те, що світ розділений на користувачів техніки Apple і ці люди користуються кабелем Lightning та людей, що користуються технікою інших виробників, в яких пристрої оснащені micro-USB. Використання саме micro-USB дає користувачу можливість з легкістю знайти провід для зарядки свого пристрою, а на момент розробки полегшити роботу з керуючим пристроєм. На етапі розробки були проведені тести з mini-USB. Цей тип показав меншу швидкодію та дістати цей провід сьогодні вже досить складно, тому було прийняте рішення пошукати більш сучасні та швидкі варіанти для обміну даними та для зарядки.

-унікальним є використання GPS на частоті в 19 Hz, за допомогою цього можна отримати досить точні дані про вимірювання.

Більшість пристроїв для вимірювання динаміки автомобіля оснащені GPS модулем та працюють на частоті всього в 10 Hz, лише VBOX Sport пропонує частоту в 20 Hz. Модуль, що використовується в дипломному проекті, фізично може показати максимальну швидкість лише в 19,51 Hz, тому було прийняте рішення використовувати модуль на частоті в 19 Hz, це дасть змогу пристрою бути кращим за більшість конкурентів та боротися з лідером ринку.

З точки зору програмного забезпечення можна виділити:

-фільтрування даних отриманих від акселерометра та гіроскопа шляхом математичних обчислень;

Дані, що надходять з акселерометра та гіроскопа, не є точними і потребують обробки їх. Був створений фільтр, що націлений на вирішення цих питань. Фільтр є математичними обчислення отриманих даних. За рахунок

цього, дані що надходять можна вибудувати в рівний графік без різких перепадів, за умови нерухомого модулю. Оскільки старт відліку відбувається з моменту росту показників перевантаження, то цей фільтр є необхідним для коректної роботи пристрою.

-обробка даних отриманих з GPS та швидке переведення їх в одиниці вимірювання швидкості;

Дані з GPS приходять в форматі довготи та широти. Слід було написати алгоритм, який максимально якісно та швидко буде обробляти отримані дані та переводити їх в одиниці вимірювання дистанції та швидкості. Оскільки історично заїзди відбувалися на відстань в 1/4, 1/2 та 1 милю, то на пристрій користувача приходять дані про швидкість, довгота та широта для подальшого обчислення та переведення даних що надходять в різні системи числення відстані. Поставлена задача була виконана та реалізована в проекті шляхом впровадження швидкого алгориту. Переведення отриманих даних в швидкість та дистанцію відбувається максимально швидко та якісно, без затримок.

-сортування отриманих даних на дійсні та не дійсні з метою відображення якості отриманого результату;

Дані що надходять з пристрою на телефон формують результат заїзду. Ці результати можуть бути відсортовані на дійсні та не дійсні. Дійсними результатами є заїзди, в яких під час заїзду висота змінювалася не більше ніж на -0,8% в залежності від відстані, яка була подолана протягом заїзду. Не дійсними є результати вище -0,8% за умови, що відстань заїзду перевищувала 200 метрів, за умови більшого схилу результати будуть визначені як не дійсні, тому що час прискорення до цієї швидкості буде меншим ніж насправді, через це було впроваджена система сортування отриманих результатів.

-архівування результатів з метою економії пам’яті пристрою користувача;

Масив даних що надходить не є дуже великим, але з часом може зайняти дуже багато місця на пристрої користувача. Якщо користувач є досить

активним, то це може стати великою проблемою. З метою економії пам’яті на пристрої користувача результати, що надходять з пристрою архівуються. Якщо користувач хоче передивитися результати своїх попередніх заїздів, то дані розархівовуються. Не архівуються лише результати, що збережені до обраних заїздів користувача. Через це результати архівуються з метою економії пам’яті користувача.

-відключення звукових оповіщень на пристрої користувача з метою підвищення безпеки під час заміру;

За статистикою більшість аварій, що відбуваються на дорозі виникають через не уважність водія, тому що водій відволікається на телефон та оповіщення, що надходять. На організацію безпеки слід уділити ретельну увагу, тому що це можу коштувати користувачу життя. Звукові оповіщення на швидкості можуть відволікти водія від дороги та призвести до аварійної ситуації на дорозі. Це не безпечно для користувача та для обставин на дорозі. Відключення звукових оповіщень відбувається шляхом включення білого шуму на телефоні користувача під час старту заїзду. Це дозволяє підвищити безпеку на дорозі.

-система безпеки запобігання MITM атак для попередження втрати результатів;

MITM(Men-In-The-Midle) атака є найстарішим та найпоширенішим видом хакерських атак. Дослівно це перекладається як «людина по середині», тобто коли злочинець вступає в ролі посередника при передачі інформації. Цей тип кіберзлочину є розповсюдженим та руйнівним. Суть атаки доволі проста: злочинець перехоплює трафік з одного пристрою та відправляє його кінцевому одержувачу, тобто користувачу, попередньо прочитавши та змінивши ці дані на свою користь. Кожного разу коли 3 сторона перехоплює трафік це можна ідентифікувати MITM атакою. Такі дії зовсім не важко зробити злочинцеві навіть без належної автентифікації. Найпопулярнішим способом реалізації цієї атаки це є фальшива точка доступу. Фактично пристрій користувача «вважає»,

що підключений до легітимної мережі, а насправді це є приладом шахрая. Такий підхід дозволяє контролювати трафік користувача. Убезпечення користувача відбувається шляхом унікального розпізнавання пристрою та приховання точки доступу після підключення до неї.

2.5 Типова реалізація.

При реалізації дипломного проекту за основу було взято GPS модуль Ublox Neo-6M. Для передачі даних та керування всією схемою було обрано ESP32, що має вбудований Bluetooth та Wi-Fi модуль. Для підвищення якості та точності вимірювань було впроваджено використання 3-х осьового акселерометра, це зумовлено похибкою у роботі GPS. Модуль MPU6050 має не лише 3-х осьовий акселерометр, але і 3-х осьовий гіроскоп, він є необхідним в проекті, тому що дає можливість пристрою виміряти зміну висоти під час заїзду, що позитивно впливає на якість результатів вимірювань динаміки автомобіля. Для реалізації автономної роботи було використано акумуляторну батарею великої ємності та контролер живлення для зарядки батареї, живлення всього пристрою та контролю за струмом в схемі.

Темою дипломного проекту є пристрій на базі GPS для вимірювання динаміки автомобіля. Логічно прийти до висновку, що головним модулем всього пристрою є GPS.

GPS модуль використовується для отримання географічних координат місця положення антени приймача в даний момент за допомогою GPS(системи глобального позиціонування) та їх передачі на керуючий пристрій. Модуль може бути використаний в приладах під керуванням Arduino, AVR, PIC, ARM. Практичне примінення: позиціонування роботів, безпілотних літальних апаратів, метеорологічних датчиків і так далі. Модуль не рекомендується використовувати в професіональних системах. Оскільки пристрій, що розробляється, є пристроєм аматорського рівня, то його використання в схемі не є критичним [13].

Для використання GPS модуля спочатку необхідно підключити його живлення до контролера. Після того як модуль отримає зв’язок з супутником, повинен загорітися, а потім почати моргати світловий діод. Після цього в пристрій можна записати бібліотеку для отримання даних з модуля та програму для роботи з ним. Модуль видає результати в форматі NMEA. Для роботи з даними, які передає модуль та для налаштування можна використовувати програму від виробника U-center.

Для закріплення модуля на платі на корпусі є 4 отвори, що одночасно і є контактами для живлення та обміну даними. Керування GPS може відбуватися або з керуючого пристрою по інтерфейсам (UART, SPI, DDC, IIC) або з комп’ютера по інтерфейсу USB. На платі модулю присутній один UART інтерфейс для підключення живлення, запису та читання даних. Мікросхема NEO-6M-0-001 дозволяє використовувати ще декілька інтерфейсів для передачі даних, але вони не виведені(їх можна підключити самостійно). Живлення на GPS модуль може відбуватися від керуючої плати, USB порту комп’ютера або зовнішнього джерела живлення. Напруга живлення 3-5 В.

Акселерометр та гіроскоп є типовим прикладом МЕМС, які є в кожному смартфоні, планшеті і так далі. В справжній час широкого поширення отримали мікроелектрономеханічні системи (МЕМС). Це стало можливо завдяки їх мініатюрності, великої функціональності, високої надійності, низького енергоспоживання та низька вартість. Акселерометр – це пристрій, що дозволяє виміряти прискорення тіла під діями зовнішніх сил. В основі цього модуля лежить мікросхема MPU6050, в якому розміщається відразу 2 датчика: акселерометр та гіроскоп. На платі вже є в наявності вся необхідна обв’язка, а також перетворювач напруги. Підключення модуля відбувається відповідно стандартній схемі для інтерфейсу I2C.

Акселерометр використовується для вимірювання лінійної швидкості, а гіроскоп для кутових швидкостей. Сумісне використання акселерометра та

гіроскопа дозволяє визначати рух тіла в трьохвимірному просторі.

Програма отримує оновлення кожні 20 мілісекунд та виводить їх в послідовний порт.

Датчик MPU6050 дозволяє налаштовувати точність вимірювань. Можна вибрати один із 4 класів точності: ±2G, 4G, 8G та 16G, де 1G – це одна земна гравітація. Використовувана бібліотека за-замовчуванням налаштовує датчик на діапазон ±8G. З іншої сторони, MPU6050 має 16 розрядний АЦП. 2 в степені 16 дає число 65536. Оскільки датчик може вимірювати від'ємні та позитивні значення прискорення, то він буде видавати числа від -32768 до +32786. Склавши ці два факти виходить, що при налаштуваннях 1G буде рівний числу 4096 (ну а -1G рівний числу -4096). Це повністю відповідає значенням графіків, що моделює програма під час роботи.

МЕМС пристрої виготовлюють на кремнієвій підкладці аналогічно технології виробництва однокристальних інтегральних мікросхем, тому їх розміри варіюються від декількох десятків мікрон до декількох міліметрів.

Існує декілька різновидів МЕМС гіроскопів, вони відрізняються внутрішньою будовою, але їх всіх об'єднує, те що їх робота основана на використанні сили Коріоліса. В кожному із них є робоче тіло, виконуюче зворотньо-поступальні рухи. Якщо рухати підкладку, на якій знаходиться це тіло, то на нього починає діяти сила Коріоліса, направлена перпендикулярно вісі обертання та напряму руху тіла.

Знаючи лінійну швидкість та силу Коріоліса можна визначити кутову швидкість.

Одна із можливих реалізацій гіроскопа має наступну структуру: закріплена на гнучких підвісках рамка, всередині якої здійснює поступальні коливальні рухи деяка маса.

Коливання робочої маси відбувається вздовж осі X та генеруються

електростатично, а коливання внутрішньої рамки можливе лише вздовж осі Y. Між внутрішньою рамкою та підкладкою розміщені обкладки плоских конденсаторів (сенсори переміщення), таким чином вимірює їх ємність, можна фіксувати рух рамки відносно підкладки.

Але коливання внутрішньої рамки можуть викликатися не лише силою Коріоліса, але і лінійним прискоренням, яке діє вздовж осі Y. Проблема вирішується шляхом розміщенням на одній підкладці двох рамок, в кожній із яких розміщена робоча маса. Обидві маси коливаються в протифазі, відповідно, в певний момент часу сила Коріоліса, діє на першу масу, направлена протилежно силі, впливає на другу. Сигнали генеровані силою Коріоліса, будуть складатися, а синфазна складова, породжена лінійним прискоренням – віднімається [8].

В якості керуючого пристрою та Bluetooth модулю виступає плата esp32. Пристрій представляє собою систему на кристалі, побудованою за технологією TSMC 40 нм, з Wi-Fi та Bluetooth контролерами. Він оснащений 2-во ядерним 32-бітним процесором, який працює на частотах 80, 160 або 240 MHz. Також в систему інтегровані антенні комунікатори, радіочастотні компоненти, фільтри, підсилювачі, модулі управління живлення. Підключається до комп'ютера ESP32 через звичайний micro-USB.

ESP32 розроблений для переносної електроніки та для додатків інтернет речей, виконаний в дуже мініатюрному корпусі, потребуючий для інтеграції близько 10-ти зовнішніх компонентів. Він оснащений чудовим функціоналом та багатообіцяючими можливостями. Сумісні в одному чіпі Wi-Fi та Bluetooth, двох процесорних ядер та багатого набору периферії робить ESP32 найкращим модулем керування для використання в даному дипломному проекті. Його потенціал та функціонал закладений розробниками є неймовірно великим та зможе покрити вимоги проекту із запасом.

Обчислювальна потужність плати є дуже високою. У ESP32 є два ядра,

кожне із яких працює на частоті 160 MHz. Друге ядро значно спрощує процес розробки. Так, наприклад, одне ядро може взяти на себе задачі реального часу по роботі зі складними завданнями, для роботи з графікою або для управління двигунами, а друге може обробляти комунікаційні протоколи та в цілому забезпечувати зв’язок. Це дозволяє розділити час між задачами. В проекті ця можливість використовується іншим чином. Оскільки пристрій не працює з графікою, то ядра можна розділити на роботу з модулями.

В пристрої керування традиційно присутнє вбудоване управління енергоживлення. Для цього використовується лінійний регулятор, індивідуальне живлення для RTC(ядро низького енергозбереження), пробудження по таймеру або сенсорному датчику.

Плата ESP32 є досить новою, вона вже активно використовується в комерційних проектах, пов’язаних з мобільними додатками, електронікою та задачами IoT.

Плата дозволяє швидко програмувати модуль завдяки вбудованому адаптеру USB-TTL. На ній також розміщено кнопки програмування та скидання, а також регулятор напруги для живлення мікроконтролера ESP32 напругою 3.3 V. Також плата дає зручний доступ до виходів модуля, крок між якими досить вузький, щоб без проблем працювати з ними. Крок між виходами плати складає 2,54 mm, що є стандартом для DIP-корпусів, з яким зручно працювати без спеціального інструменту.

Модуль зарядки оснований на чіпі TP4056 – контролері заряду літієвих акумуляторів з вбудованим термодатчиком. Цей модуль є закінченим рішенням з лінійним зарядом по принципу постійної напруги/постійний струм для одноелементних літієвих акумуляторів.

Плати передбачені для роботи з елементами 18650 (в основному через зарядний струм в 1А), але при перепайці резистора можна використовувати для любих акумуляторів на 3.7 V.

Контролер виконаний в корпусі SOP-8, має на нижній поверхні металевий теплообмінник не під’єднаний з контактами, дозволяє заряджати акумулятор током до 1000ма (залежить від резистора, що видає струм). Потребує мінімум навісних компонентів.

Під час розробки додатку для iOS слід було організувати обмін даними між апаратним забезпеченням та телефоном або планшетом користувача. Передача даних відбувається за допомогою Bluetooth. З метою використання сучасних технологій та економії заряду батареї телефону та приладу було вирішено використати Bluetooth 4 покоління, який має протокол BLE.

BLE є широко вживаним протоколом з наднизьким енергоспоживанням та радіусом дії до 10 метрів. Є новою специфікацією Bluetooth і дає підтримку широкому діапазону додатків та зменшує розмір кінцевого пристрою для зручного використання.

Три частоти виділені для широкомовних безадресних посилок, а всі інші для передачі пакетів при встановлені логічних каналів зв’язку між пристроями. Відомою особливістю Bluetooth є те, що при передачі пакетів кожний наступний пакет передається на іншій частоті, що вибирається псевдовипадково зі списку дозволених.

Всі дані в пакетах BLE можуть кодуватися та перекonуватися. Також використовується динамічна випадкова генерація адрес пристроїв та їх ідентифікація з використанням хешування, тобто, переловивши в ефірі адресу пристрою зловмисник не зможе використовувати її довше ніж 15 хвилин, оскільки адреса за цей час зміниться за невідомим для всіх алгоритмом.

Модулі BLE мають змогу працювати як однонапрямлений маршрутизатор, тобто без встановлення двонапрямленого з’єднання, просто транслювати до ефіру якісь дані в формі пакетів оголошень, наприклад, температуру. Телефон або планшет можуть приймати дані з сотень таких передавачів без зайвих попередніх дій по пошуку, встановлення з’єднання,

вводу паролі та іншого.

Іншою можливістю передавати дані без встановлення зв’язку каналу є передача в режимі запит-відповідь. Цим BLE суттєво відрізняється від Wi-Fi, де навить для найпростіших дій треба встановлювати з’єднання.

З’єднання додатку з пристроєм відбувається за допомогою бібліотеки Core Bluetooth.

За великим рахунком Core Bluetooth є лише обгорткою, тому що протоколи BLE самі по собі досить складні. Це було зроблено компанією Apple спеціально, щоб це було більш зручнішим та легшим у використанні розробниками.

Щоб користувач не забував свій пристрій слід визначати наскільки телефон або планшет користувача близько до приладу. Це зробити дуже складно і нормальної документації для цього також нема. Компанія Apple значно спрощує цей механізм за допомогою iBeacons, він відразу дає дані про те, наскільки пристрої поруч. Але для роботи зі сторонніми пристроями нема настільки просто способу. Складно зрозуміти, знаходиться пристрій в доступному діапазоні чи ні. Інколи при пошуку приладів він може дати знати про себе лише один раз та пропасти, тоді спроба підключення буде безуспішною.

Для цього слід використати наступний спосіб: зберігати стек з мітками дати та рівня сигналу (RSSI), оскільки ми працюємо з GPS, то є можливість зберігати навіть координати кожного повідомлення, отриманого в спеціальному класі, створеному для цього. Спосіб є досить схожим на те, як CoreLocation зберігає мітки часу на відповідні координати. Зазвичай, чим більший сигнал RSSI – тим ближче пристрій. Зрозуміло що, знаходиться прилад в доступному діапазоні чи ні – складніше, оскільки це поняття є досить неоднозначним. Для цього треба використовувати усереднене значення сигналу. У підключеного пристрою рівень сигналу слід запитувати вручну

щоразу, коли буде необхідність його дізнатися.

Для зберігання отриманих результати слід використовувати базовий фреймворк Core Data. Core Data – це фреймворк, який керує та зберігає дані в додатку. Можна розглядати Core Data, як оболонку над фізичним реляційним сховищем, преставляючи дані у вигляді об’єктів, при цьому Core Data не є базою даних.

Для створення сховища в додатку використовуються класи NSPersistentStoreCoordinator або NSPersistentContainer.
NSPersistentStoreCoordinator створює сховище вказаного типу на основі моделі, можна вказати розміщення та додаткові опції. NSPersistentContainer дає можливість створення з мінімальною кількістю програмного коду.

Працює це наступним чином: якщо по вказаному шляху існує база даних, то координатор перевіряє її версію та, за необхідністю, робить міграцію. Якщо бази не існує, то вона створюється на основі NSManagerObjectModel. Щоб це все запрацювало правильно, перед внесенням змін в модель слід створити нову версію в Xcode.

Якщо користувач швидко вийде з програми та перезавантажить свій телефон або планшет, дані з програми можуть пропасти. Core Data націлена вирішувати такі питання. Всі дані користувача можуть зберігатися більш надійно та вони будуть збережені, навіть якщо пристрій буде перезавантажено. Це позитивно впливає на безпеку даних та надійність додатку.

З точки зору безпеки є можливість не зберігати дані в файловому вигляді, але при цьому використовувати кешування впродовж сесії та дані у вигляді об’єктів, для цього можна використати сховище типу «In-Memory». Власне не забороняється мати в одному додатку декілька сховищ різних типів.

В Core Data операції можуть виконуватися в своїй транзакції, для рішення цієї проблеми є досить простий спосіб – від’єднання дочірнього контексту [10].

3. РЕАЛІЗАЦІЯ ПРОЕКТУ

3.1 Вибір схеми моделі

При проектуванні дипломного проекту було відібрано та проведено тести багатьох схем підключення модулів. Кожний з варіантів мав свої переваги та недоліки, але необхідно було вибрати найкращий варіант не зважаючи на її недосконалість. Ключовим питанням стало максимально зручне підключення та можливість «гарячої» заміни модулів. Це дозволяє швидко замінювати модулі, які було вирішено замінити на кращі або якщо якийсь пристрій вийшов з ладу, то також його замінити. Всі підключення не є випадковими, а відповідають вимогам проекту та правилам роботи з комп'ютерною електронікою. В кінцевому варіанті обрана саме така схема моделі пристрою (див. додаток 1).

Такий вибір контактів обумовлений зручністю підключення та особливостями контактів, що значною мірою впливає на швидкодію та завадостійкість пристрою.

Із акумуляторної батареї виходить 2 мідні проводи живлення, що мають бути підключені до проміжного пристрою керування напруги або напряму до плати, що не є добре. Підключення на пряму до плати може бути не безпечним через то що у випадку перенапруги може «згоріти» весь пристрій або батарея критично розрядиться і втратить свою ефективність, що негативно вплине на подальше користування всім пристроєм.

На контролері заряду є micro-USB роз'єм для зарядження батареї, а також 6 контактів:

-B+ для плюсового контакту батареї;

- B– для мінусового контакту батареї;

- OUT+ вихід плюсового контакту на керуючий пристрій;

- OUT– вихід мінусового контакту на керуючу плату;

- IN+ плюсовий вхід на контролер живлення для зарядки;

- IN– мінусовий вхід на контролер живлення для зарядки.

При розробці було задіяно лише 4 контакти, які власне і були розпаяні для використання:

- B+ для плюсового контакту батареї;
- B– для мінусового контакту батареї;
- OUT+ вихід плюсового контакту на керуючий пристрій;
- OUT– вихід мінусового контакту на керуючу плату.

Заряджувальний пристрій підключається до 2 контактів на керуючій платі: VIN(voltage input) та GND (ground). Це спеціальні контакти передбачені для підключення зовнішнього живлення. Однак, слід зазначити, що при підключенні керуючого пристрою до ноутбуку струм, що протікає в колі є вищий ніж коли плата живиться лише від акумуляторної батареї.

Акселерометр та гіроскоп має в корпусі 8 контактів для підключення:

- VCC – позитивний контакт живлення;
- GND – земля;
- SDA – лінія даних I2C;
- SCL – лінія синхроімпульсів I2C;
- INT – налаштовані переривання;
- AD0 – I2C адрес. За замовчуванням AD0 підтягнутий до землі, тому адреса пристрою – 0x68, якщо підключити AD0 до контактів живлення, то адреса зміниться на 0x69;
- XCL, XDA – додатковий I2C інтерфейс для підключення зовнішнього монітору.

В дипломному проекті використовується тільки 4 контакти:

- VCC – позитивний контакт живлення;
- GND – земля;
- SDA – лінія даних I2C;
- SCL – лінія синхроімпульсів I2C;

Для передачі необхідної інформації достатньо лише 2 контакти, що забезпечують високу швидкодію та безвідмовну роботу пристрою.

Інші контакти не потрібні для роботи в цьому проекті, тому що зовнішній монітор не буде підключатися в подальшій роботі пристрою, а робота з перериванням не буде впроваджена в дипломному проекті, змінювати адресу пристрою нема необхідності, оскільки в розробці задіяний лише один акселерометр з гіроскопом.

Контакти, що необхідні для передачі даних підключені до пристрою керування через цифрові контакти D21 та D22. Плата керування лише отримує дані з модулю і нічого не передає на неї. Контакти, що відповідають за живлення підключені до контактів керуючої плати 3V3 та GND відповідно.

GPS модуль складається з процесору, 3 транзисторів, резистору, батареї, світлового діоду, 4 контактів та роз'єм для підключення антени.

GPS оснащений такими контактами для підключення:

- VCC – позитивний контакт живлення;
- GND – земля;
- RX – читання інформації з модуля;
- TX – запис інформації на модуль;
- Роз'єм для підключення зовнішньої антени.

В розробці використовуються наступні контакти:

- VCC – позитивний контакт живлення;
- GND – земля;
- RX – читання інформації з модуля;
- Роз'єм для підключення зовнішньої антени.

Оскільки відсутня необхідність в запису інформації на GPS, то підключення до TX не потрібне для роботи з модулем. Для налаштування роботи GPS на необхідній високій частоті TX використовується. Тому що запис відбувається за допомогою USB-TTL пристрою.

Модуль оснащений додатковою батареєю для більш швидкого «холодного» старту роботи пристрою. Але на практиці це не працює і для «холодного» старту необхідно більше однієї хвилини на вулиці, а не 20 секунд,

як заявляється виробником. В приміщені з товстими стінами модуль може взагалі не включитися.

Контакти для передачі даних підключені до TX0 та RX0 відповідно на керуючому пристрої. Це є необхідно для забезпечення високої швидкодії та безвідмовної роботи пристрою. Контакти, що відповідають за живлення підключені до контактів керуючої плати 3V3 та GND відповідно.

Антена підключена до плати GPS модуля через коаксіальний кабель. Сама антена схована в керамічний корпус. Це забезпечує надійність в експлуатації, тому що у разі падіння пристрою антена не розіб'ється.

3.2 Вибір інструментального програмного забезпечення

Перед розробкою програмного забезпечення слід визначитися з інструментарієм для написання програмного коду.

Розробка програми відбувається в 2-етапи:

- Створення програмного забезпечення для пристрою;
- Написання додатку для iOS.

З точки зору написання програми для роботи апаратного забезпечення вибір є досить очевидним, оскільки інших інструментів для роботи з платою керування не передбачено, відповідно в якості середовища розробки було вибрано Arduino IDE, а в якості мови програмування змінений C/C++.

Мова програмування пристроїв Arduino оснований на C/C++ та скомпонований бібліотекою AVR Libc і дозволяє використовувати любі її функції. Він простий у вивченні, на даний момент Arduino – це, напевно, найзручніший спосіб програмування пристроїв на мікроконтролерах. Мову Arduino можна розділити на 4 розділи: оператори, дані (змінні та константи), функції та бібліотеки.

Мікроконтролер Arduino програмується на спеціальній мові програмування, основному на C/C++. Мова програмування Arduino є

різновидом C++, іншими словами, не існує окремої мови програмування для Arduino.

В Arduino IDE всі написані скетчі компілюються в програму на мові C/C++ з мінімальними змінами. Компілятор Arduino IDE значно спрощує написання програми для цієї платформи і створення пристроїв на Arduino стає набагато доступніше людям, не маючих великих знань в мові C/C++.

Програмування на Arduino значно простіше та зрозуміліше за мову C/C++. На мові програмування Arduino включає в скетчі послідовний порт на швидкості 9600 біт в секунду.

При виборі мови програмування для написання додатку для iOS все значно складніше, компанія Apple пропонує на вибір стару мову програмування Objective-C та їхню особисту нову мову Swift.

В червні 2014 року в світі Apple сталось те, чого не очікував ніхто: компанія Apple представила нову об'єктно-орієнтовану мову програмування – Swift, яка прийшла на зміну Objective-C, використовувану раніше для розробки додатків для OSX та iOS.

В суспільстві таке рішення тепло зустрліли та з великим ентузіазмом. Це не дивно, тому що відразу після появи Swift з'явились слухи про його простоту, доступність та зручність, що є дуже важливо для багатьох проектів.

Мова Swift за задумом його виробників створена не лише замінити Objective-C в якості базової мови для додатків iOS, але і замінити C в усіх сферах, в тому ж числі IoT. Але якби Swift був би насправді таким чудовим, як про нього говорять, то давно б зайняв місце на вершині рейтингів. Але поки що це не так. Для порівняння слід розглянути плюсі та мінуси кожної мови.

Objective-C

Короткий опис: мова, створена на початку 80-х років минулого віку, шляхом покращення C з популярним на той час Smaltalk. Спочатку

сприймався, як просте надбудова, модифікуюча деякі синтаксичні конструкції, але після того, як за ліцензування взялась спочатку компанія NeXT, а потім на правах приємника і Apple, Objective-C став одним із найбільш популярних мов для розробки додатків.

Плюси:

1. Динамічна типізація. В деяких випадках це дійсно може стати ключовою перевагою. Наприклад, спрощує створення не складних програм.

2. Документація та суспільство. Більше 20 років успішного примінення мови посприяло появи великої кількості якісних ресурсів та книг. Сьогодні кожний, хто бажає вивчати Objective-C, без труднощів знайде відповідь на питання, що цікавить на просторах інтернету.

3. В порівнянні з C++, та і з багатьма іншими мовами того часу, Objective-C надає розробнику куди більше гнучкості.

4. Відносно простий синтаксис. Простота полягає в першу чергу в розумінні алгоритмів і того, як працює виконувана машина.

Мінуси:

1. Низький рівень читання коду, що затрудняє вивчання мови новачкам.

2. Динамічна типізація передбачає можливість появи (пропуску) помилок навіть під час компіляції.

3. Обмежена функціональність. Досить логічно, що мова створена в минулому віці, отримала перші оновлення в 21 віці не буде мати багатий функціонал.

4. Не найвища продуктивність, знову ж таки викликана динамічною типізацією.

Swift

Короткий опис: в 2014 році спеціалістами Apple було представлено світу

заміну звичному Objective-C. Серед цілей, що були наслідувані були заявлені високий рівень читання коду та стійкість до помилок розробника.

Плюси:

1. Для написання додатку треба менше коду, хоча б просто тому, що тут реалізований спрощений принцип роботи з рядками та заявами, що повторюються.

2. Зручний для читання. Стандартна перевага кожної нової мови.

3. Більше можливостей в порівнянні з Objective-C, особливо в можливістю управління пам’яті.

4. Повноцінна взаємодія з кодом, написаному на Objective-C.

5. Підвищена безпека. Це виявляється в обробці вказівників і у вимогливості компілятора, і в тому, що в саму компіляцію можна вбудувати опціональну змінну nil для забезпечення зворотнього зв’язку.

Мінуси:

1. Через молодості мови і не переведених на Swift кодів OSX та iOS необхідні хоча б мінімальні знання Objective-C.

2. Компілятор видає зайве і просто збиваюче з толку помилки, які розробнику, що прийшов з іншої мови здається, як мінімум, не звичайним.

Не дивлячись на деякі переваги у мові Objective-C і схильність досвідчених розробників ідеалізувати його, хоча б від тотального яблучного господарювання Swift відділяє час та кропітка робота розробників над його недоліками. Тому, що досить логічно, що мова, що з’явилась світу всього пару років тому повинен пройти не просту стадію народної критики та допрацювань перед тим як піднятися на вершину. А Objective-C поступово піде в історію, залишив про себе лише добру пам’ять.

Остання версія Swift 4.1 вийшла в березні 2018 року. До цього моменту

у версіях 2.0 та 3.0 мова програмування отримала підвищену продуктивність, покращуючи синтаксис мови та підтримку операційних систем OSX, iOS і Linux.

Мова програмування від Apple залишається одним із найпотрібніших у світі. Не дарма релізи додатків та ігор в першу чергу виходять саме на пристрої під управлінням iOS. Крім того Swift є лідером в списку GitHub по кількості публікацій вихідного коду.

Swift є найбільш популярною мовою серед новачків розробників, що щойно прибули на нову платформу, які ще не вклали час та сили в Objective-C. Хоча при цьому більшість розробників досить нормально сприймають перехід на Swift. Це є більш питанням нового зручного синтаксису та розумінні, а не швидкодії та безпеки [4].

Слід відмітити, що поріг знань для входу в роботу з Swift набагато нижчій ніж в Objective-C. Це обумовлено досить простим синтаксисом та наглядною роботою технологій, що використовуються при розробці програмного забезпечення на мові програмуванні Swift.

Отже, в якості мови програмування було обрано Swift тому, що ця мова є новою, забезпечує високу продуктивність, безпеку та простий у вивченні та роботі синтаксис.

Окрім вибору мови програмування слід визначитися з середовищем розробки. Для розробки мовою програмування Swift існує 2 середовища розробки. Особисте середовище xCode від Swift та AppCode від JetBrains.

AppCode є дуже зручним середовищем, з високою швидкодією та стабільною роботою в порівнянні з xCode. Але Apple поширює своє програмне забезпечення безкоштовно в порівнянні з JetBrains.

Перед початком роботи з AppCode слід здійснити багато налаштувань для подальшого виконання необхідної роботи розробника. В xCode все

налаштовано завчасно і розробнику не потрібно витрачати свій час на впровадження цих налаштувань.

Ключовим недоліком AppCode є те, що для роботи з графічним інтерфейсом дизайнера він вимушений викликати його з xCode, а також в ньому відсутні симулятори, для цього також потрібно встановлювати та запускати xCode.

Тож вибір середовища розробки є очевидним. Слід зазначити, що компанія Apple рекомендує використовувати Swift у разом з xCode, тому що він дозволяє використовувати всі необхідні фреймворки та технології.

3.3 Опис архітектури та принцип роботи.

Перед початком розробки слід похвилюватися про архітектурний патерн для написання програми. Якщо це не зробити, то виникне проблема, через те що відлагоджуючи величезний клас з десятками різних методів та властивостей, можливості найти та виправити помилки може не виявитися. Звісно такий клас буде складно тримати в голові як єдине ціле, тому завжди буде втрачатися із виду якась важлива деталь. Якщо така проблема вже виникла, то це дуже ймовірно, що:

- Цей клас – наслідник UIViewController;
- Дані зберігаються прямо в UIViewController;
- UIView-підкласи ні за що не відповідають;
- Model – просто контролер для даних;
- Юніт-тести не виконувались.

Слід дати визначення добрим проявам архітектури:

- Збалансоване розприділення обов’язків між сутностями з жорсткими ролями;

Розприділення зменшує навантаження на керуючий контролер, коли слід

вияснити, як працює та чи інша сутність. Чим більше буде розвиватися структура, тим легше буде адаптація до складної концепції. Але у всього є свої обмеження і вони досягаються досить швидко. Таким чином найпростішим способом полягає в розділенні обов'язків між кількома сутностями по принципу єдиної відповідальності.

- Тестування. Зазвичай виходить із першої властивості (це легко досягти при відповідній архітектурі);

Тестування архітектури програми визначає, наскільки легко буде написати юніт-тести, а частіше всього – чи буде можливість їх писати взагалі і чи варто їх писати. Як правило, юніт-тести слід виконувати після оновлення функціоналу або після рефакторінгу якихось тонкостей класу. Це означає, що тести зберегли розробника від виявлення проблем в «рантаймі». Це могло б статися вже не пристрої користувача, а розробник міг про це навіть не знати і виправити не вчасно.

- Простота використання та низька ціна обслуговування.

Чим менший код – тим нижча ймовірність появи помилок. Вибравши правильну архітектуру розробник має можливість суттєво зменшити об'єм написаного коду. Це повпливає на швидке виправлення помилок та меншу ймовірність їх появи в процесі розробки проекту. Що є безумовно важливим фактором

Для роботи з проектами Apple рекомендує використовувати їх базовий патерн MVC. Що має високу наглядність архітектури програми та зручність для початківці при створенні додатків.

Традиційний MVC відрізняється від Apple MVC. В традиційному View не зберігає стан в собі. Controller просто «рендерить» View при зміні Model. Наприклад, веб-сторінка повністю перезавантажується після того, як користувач натискає на посилання для переходу на інше місце. Хоча можна реалізувати традиційний MVC в середовищі iOS, це не має важливого сенсу

через архітектурні проблеми: всі три сутності тісно пов’язані, кожна сутність знає про двох інших. Це сильно зменшує можливість повторного використання кожного з елементів.

В Apple MVC Controller є посередником між View та Model, відповідно два останні не знають про існування одне одного. Тому Controller складно повторно використати, але це не є критично, тому що потрібне місце для гнучкої бізнес-логіки, що не вкладається в Model.

Проект з досить складним функціоналом: робота з Bluetooth, відображення даних в реальному часі, збереження старих даних, профіль користувача. У висновку виходить досить громіздкий контролер, який вирішує вище зазначені проблеми та кучу коду, який не можна перевикористовувати. Іншими словами, MVC може стати великою проблемою в процесі розробки та при додатковій підтримці проекту. Це не може забезпечити високу модульність та перевикористовування [6].

В якості альтернативи для MVC слід розглянути архітектури MVVM, VIPER. На даний момент ці архітектури є найбільш популярними архітектурними рішеннями, що використовуються при розробці крупних додатків, що потребують участі в розробці великих команд, що є добре тестованими, мають тривалу підтримку та постійно розвиваються. Ці архітектури хоч і відрізняються, але не являються унікальними. MVVM говорить, що окрім View та Model повинен бути ще шар ViewModel. Але нічого не говориться про те як повинен формуватися цей шар, але і про те, як відбувається запит даних – відповідальність для цього шару не визначена чітко. Існує велика кількість способів його реалізації і є можливість використовувати кожний із них.

З іншого боку VIPER є унікальною архітектурою. Він складається із великої кількості шарів, кожний із яких має досить визначену зону відповідальності і менше ніж MVVM підвергається впливу розробників.

Коли йде мова до вибору архітектури, то єдиного правильного рішення зазвичай не існує, але слід врахувати декілька важливих факторів. Якщо проект великий та тривалий, з чіткими вимогами і необхідно мати широку можливість перевикористання компонентів, то VIPER буде не кращим рішенням. Властивість MVVM до більш чіткого розмежування відповідальності дає можливість краще організувати тестування та кращу продуктивність програми, що розробляється.

При розробці програмного забезпечення було вибрано архітектуру MVVM(Model-View-ViewModel) для більш зручної роботи.

В iOS одним із найпопулярніших патернів, з якого починають всі новачки є MVC(Model-View-Controller). Оскільки дипломний проект є досить громіздким, то контролер починає брати на себе все навантаження.

Не зважаючи на те, що Apple рекомендує працювати з архітектурою MVC було вирішено відмовитися від цього, тому що при використанні цього патерна є змішана відповідальність за роботу, що призводить до появи певних помилок.

В MVC контролер може спілкуватися як і з моделлю, так і з представленням. В MVVM діє трошки інша схема, її дуже важливо запам’ятати: User → View → ViewController → ViewModel → Model. Тобто, користувач бачить кнопку, натискає на неї, а далі навантаження бере ViewController, виконуючи певні дії з інтерфейсом, міняє свій колір кнопки. Далі, ViewModel відправляє запит серверу на отримання даних, добавляє їх в Model або виконує якісь інші дії з моделлю.

Найголовніше, що слід запам’ятати: в MVVM з’являється новий клас ViewModel, який сам спілкується з моделлю, тобто було знято з контролера цей обов’язок і тепер контролер займається тим, чим необхідно – він працює з представленням і навіть не знає про існування моделі.

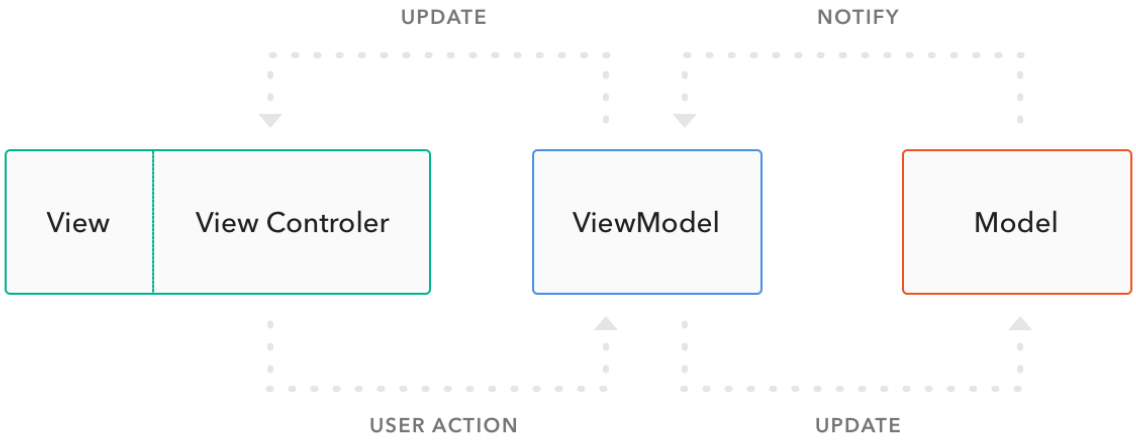

Рисунок 3.2 – відображення схеми роботи архітектури MVVM

MVVM відрізняється логікою розподілення відповідальності між модулями [7].

- **Model:** цей модуль не відрізняється від аналогічного в MVC. Він відповідає за створення моделі даних і може вміщати в себе бізнес-логіку. Також можна створювати допоміжні класи, наприклад, такі як manager-клас для управління об'єктами Model та network manager для обробки мережевих запитів та парсингу.
- **View:** саме тут і проявляються всі зміни. Модуль View в MVVM охоплює інтерфейс(підкласи UIView, файли .xib і .storyboard), логіку відображення(анімація, відрисування) і обробку користувацьких подій(натискання кнопки і так далі). В MVC за це відповідає View та Controller. Це означає, що інтерфейси(views) не залишаться не змінні, в той час як ViewController буде вміщати малу частину того, що було в ньому в MVC, та відповідно, сильно зменшиться.
- **ViewModel:** Це тепер те місце, де буде міститися більша частина коду, яка до цього була у ViewController. Шар ViewModel запитує дані у Model(це може бути як запитом до локальної бази даних, так і до мережевого запиту) і передає їх назад до View, уже в тому форматі, в

якому вони будуть там використовуватися та відображатися. Але це двонаправлений механізм, дії та дані, що вводяться користувачем проходять через ViewModel та оновлюють Model. Оскільки ViewModel слідкує за всім, що відбувається, то корисно використовувати механізм прив'язки між цими двома шарами.

Слід погодитися з тим, що UIViewController є головним компонентом iOS SDK від Apple і всі дії запускаються та створюються всередині цього об'єкта. Не дивлячись на назву, це більше, ніж класичний контролер (або презентер) із MVC (або MVP), через появу зворотних викликів, таких як viewDidLoad, viewWillLayoutSubviews та інших пов'язаних з ним методів. Саме тому слід ігнорувати ключові слова Controller в імені та використовувати його як View, де роль реального контролера приймає ViewModel.

ViewModel – повне представлення даних view. Кожний View повинен містити тільки один екземпляр ViewModel. Як правило, ViewModel використовує диспетчер для винесення даних та інвертує його в необхідний формат.

ViewModel витягує елементи через DataManager та поміщає їх в середині деякої змінної. Він також має відомості про помилки та refreshing state, що дає можливість створювати користувацький інтерфейс зі всіма необхідними умовами [15].

ItemViewController представляє список елементів в UITableView. Він містить екземпляр ViewModel та просить його отримати дані в колбек від viewDidLoad. Також передається замикання(closure), який перезавантажує UITableView, як тільки дані будуть отримані. Методи UITableViewDataSource також використовують ViewModel для отримання кількості комірок.

Архітектуру MVVM не можна розглядати без реактивного біндингу, тому що реактивні прив'язки є головною ідеєю архітектури MVVM. Вони направлені на покращення швидкої програми та завадостійкості.

Реактивні прив’язки між Model та ViewModel є головним принципом архітектури MVVM, де розробники можуть працювати з кодом ViewModel, а з дизайном можна працювати з View Interface в Designer.

В iOS SDK нема можливості біндингу, тому слід використовувати замикання. В реальних додатках можна використовувати деякі популярні розширення FRP, такі як ReactiveCocoa, RxSwift або Bond. В проекті використовувалася бібліотека Bond через його відносну простоту при роботі над додатком.

Для контролю над змінами було використано клас реактивного програмування ItemViewModel [2].

Слід детально розглянути метод bindViewModel: тут відбувається прив’язка ViewModel до View. Кожного разу, коли відбувається оновлюване значення, воно встановлює властивість isAnimating об’єкта ActivityIndicator. Або коли елементи змінені, UITableView перезавантажується. Як можна помітити в більшості випадків реактивні прив’язки полегшують код.

Інколи при наявності складних View з деякими джерелами даних. Наприклад, профіль користувача в програмі, де є інформація про користувача та фотографія профілю користувача.

Добрим прийомом буде розділити цю логіку на дві або більше ViewModel. Але існує правило: кожний вид повинен мати лише один ViewModel. В такому випадку є кращим варіантом використовувати композицію ViewModel.

Існує UserProfileViewModel, який містить дві інші ViewModel та збирає дані із них. Також існує refreshing stats внутрішніх ViewModels. Важливий момент є те, що ViewModel формує дані в необхідну форму. Перегляд просто повинен прив’язати компоненти до ViewModel та показати дані як вони є.

ViewModel – це дійсно добрий та простий спосіб передачі логіки

представлення другому об’єкту, який допомагає уникнути великих View-Controller, спростити та охватити код юніт-тестами. Саме так і отримується проста та протестована архітектура.

Спостерігаючи за виконаною програмою, можна переконатися, наскільки менше рядків програмного коду було необхідно для написання даного додатку для курсового проекту і наскільки більше було б необхідно написати, використовуючи патерн MVC.

Але після почтку використання MVVM, це не означає, що слід його скрізь використовувати його скрізь, все повинно бути логічно та послідовно. Не варто забувати, що головне – це написання гарного коду програми, який легко перевірити та легко зрозуміти. Завдяки цьому зменшується можливість виникнення помилок. Це забезпечує високу завадостійкість та безвідмовність програмного додатку.

3.4 Опис інтерфейсу додатку.

При створенні інтерфейсу для дипломного проекту було прочитано декілька книг по створенню UI інтерфейсу [12]. В дизайні увага була приділена також історичним аспектам автоспорту. Головною вимогою до графічної оболонки користувача є:

- Високий рівень наочності;
- Легкість в освоєнні новим користувачем;
- Унікальний дизайн;
- Висока швидкодія;
- Завадостійкість.

Вперше відкривши програму користувача зустрічає екран підключення до пристрою:

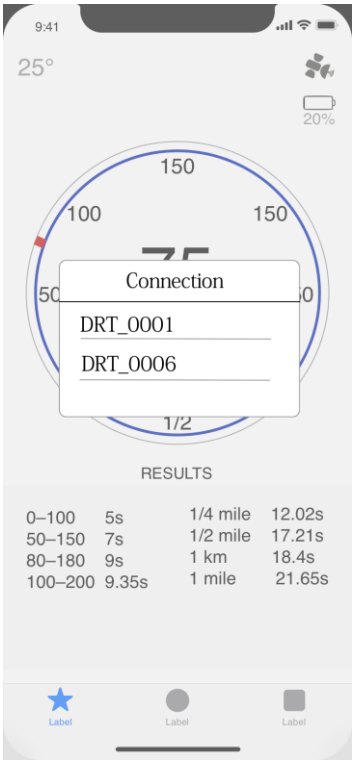

Рисунок 3.3 – підключення до пристрою DRT.

Програма сама відфільтровує пристрої з якими можливе встановлення Bluetooth з'єднання та відображає лише пристрої DRT. Цим самим користувачу не потрібно шукати свій прилад серед багатьох інших. При повторному входженні в програму з'єднання встановлюється автоматично, оскільки програма запам'ятала адресу Bluetooth модуля. Якщо телефон та обладнання поруч, але з'єднання не встановлюється, то пристрій необхідно зарядити.

Після під'єднання до DRT користувач попадає на головний екран, де відображається спідометр, температура навколишнього середовища, заряд батареї приладу у відсотках, наявність підключення до GPS та швидке відображення результатів базових замірів динаміки. Цей екран є головним екраном користувача з який він буде взаємодіяти протягом 85% часу користування додатком.

Оскільки все більшої популярності набирають OLED дисплеї, то для подальшого розвитку програми можна зробити заміну білого фону на чорний. В OLED дисплеях чорний колір досягається шляхом відключення пікселів, що

зафарбовані чорним. Цим самим можна досягти економії батареї телефона [5].

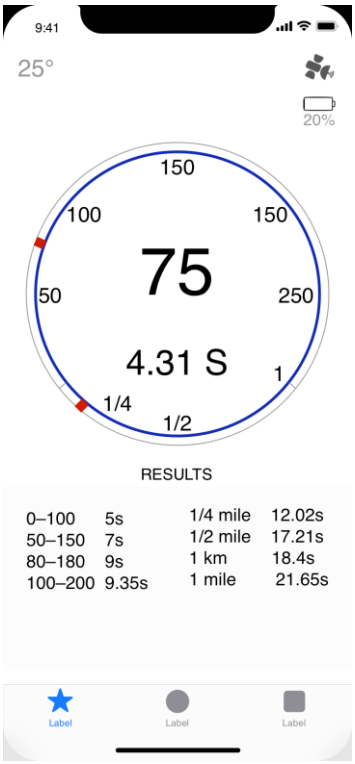

Рисунок 3.4 – спідометр.

Вище середини екрана розміщений сумісний спідометр та одометр. Верхні шкали (50, 100, 150, 150, 250) показують швидкість та вимірюються в кілометрах на годину. Червоний повзунок є «стрілкою» і змінюється в залежності від швидкості автомобіля. Нижні шкали (1/4, 1/2, 1) вказуються на пройдену дистанцію та вимірюються в милях. На одометрі також присутній повзунок і він перестає рухатися після пройденої 1 милі з моменту руху автомобіля. В середині розміщений показник швидкості та час з моменту початку заміру динаміки.

Натиснувши на поле з результатами користувач може перейти до розгорнутого звіту замірів. Вибравши окремо замір користувач попадає до наступного екрану.

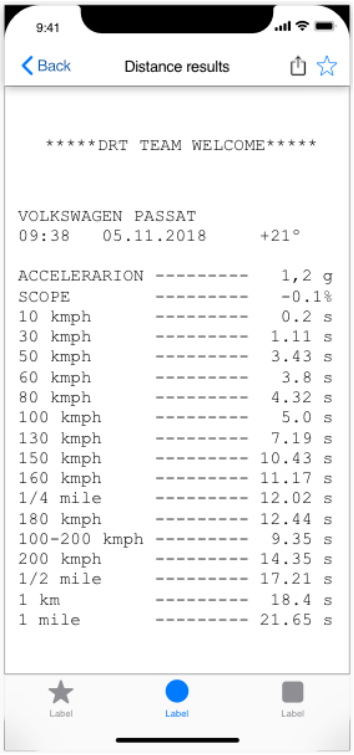

Рисунок 3.5 – екран результатів.

Дизайн цього екрану відсилається на історію драг-рейсінга. Звіт про швидкості виконаний в стилі талончика, що друкується на чековому папері, який видається після кожного заїзду на драгстріпі. На талоні відображена назва машини, час та дата заїзду, температура навколишнього середовища, перевантаження під час прискорення, зміна висоти під час заїзду, час розгону від 0 до вказанові вище швидкості та час проходження певної дистанції для формування повного розуміння про заїзд, щоб бачити в якій зоні автомобіль їде краще, а в якій гірше. Результат може бути збережений в .txt файл для поширення між друзями. Заїзд можна добавити до обраних, щоб мати швидкий доступ до результатів [1].

Для фільтрації результатів користувач має змогу добавляти свій автомобіль. Оскільки програма є актуальною для тюнінг ательє, власників автомобілів, що пройшли певні допрацювання потужності, у користувача є змога задати потужність автомобіля самому. Також для автомобіля що пройшли механічні та програмні допрацювання існує поле для розповіді про

свій автомобіль та роботи, що були виконані з ним. Саме тому для подальшого розвитку програми передбачений опис автомобіля.

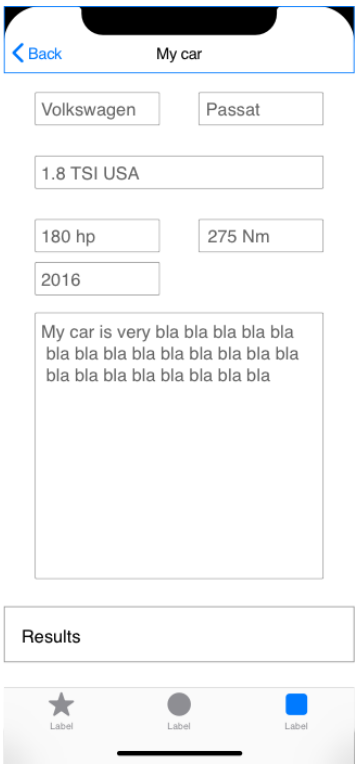

Рисунок 3.6 – екран опису автомобіля.

В залежності від автомобіля користувача, в нього можуть виникнути свої вимоги для замірів. Оскільки у користувача може бути слабкий двигун, то заміри 100-200 км/год не є актуальними, тому що автомобіль фізично не зможе досягнути такої швидкості. Так само для потужного автомобіля на моноприводі заміри 0-100 км/год можуть бути не потрібними, тому що весь крутний момент «піде» в букс. Саме тому був створений наступний екран [9].

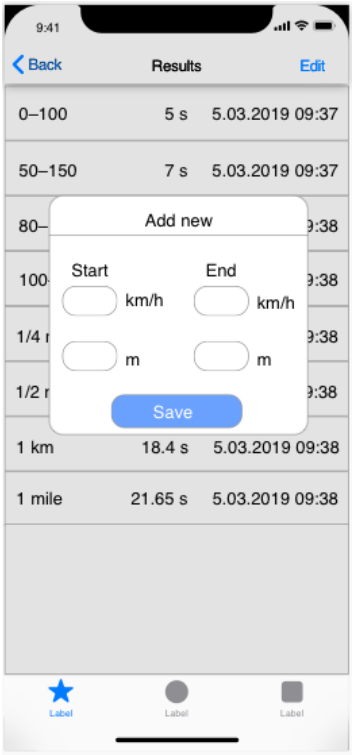

Рисунок 3.7 – екран додавання нових замірів.

На Рис. 3.7 зображено екран додавання нових замірів. Це може бути як вимірювання швидкості, так і заміри відстані. Кастомні вимірювання швидкості можуть більше підійти для користувача ніж базові метрики.

Також були створені екрани:

- Вибрані заїзди;
- Екран автомобілів;
- Екран вибору метрик;
- Екран входу користувача;
- Екран реєстрації користувача;
- Профіль користувача;
- Екран результатів;
- Перелік всіх заїздів;
- Результати заїзду обраного автомобіля.

4. ВИСНОВОК

В ході виконання дипломного проекту було досліджено як працюють любительські та професіональні пристрої для вимірювання динаміки автомобіля, проаналізовано ряд вже існуючих пристроїв та методів виконання.

У першому розділі було було проаналізовано загальний опис проблеми, готові рішення, що існують. Дослідження оказали, що проблема з покупкою та роботою таких пристроїв існує та досить актуальна в наш час. Були розглянуті основні переваги та недоліки пристроїв, що вже давно існують та є дуже популярними, але якимось чином повністю не вирішували нашу проблему. Але аналіз показав, що не одне з цих рішень повністю не вирішує проблему, що ставилась в дипломному проекті.

Другий розділ дипломного проекту було присвячено особливостям розробки пристрою, а саме: крім того описано принцип роботи створеного приладу враховуючи всі його модифікації впродовж створення, як вони працюють, основні переваги та недоліки кожної конфігурації. В даному розділі вибрано кінцеву схему-модель та описано основні особливості створеного приладу.

В останньому розділі було описано весь процес розробки програмного та апаратного забезпечення. Крім того описано принцип роботи самого пристрою та додатку, взаємодії основних модулів та відображення роботи апаратного забезпечення в інтерфейсі користувача.

Поставлена задача була виконана успішно та має право на подальший розвиток проекту. Даний проект вже може бути впроваджений як комерційний продукт. Варто зазначити, що апаратне та програмне забезпечення не підлягає кастомізації.

5. СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Артемій Лебідєв Ководство., 2009.

2. Василь Усов Swift. Основи розробки додатків під iOS та macOS., 2018.

3. Віктор Петін Проект з використанням контролера Arduino., 2015. – (БХВ Петербург).

4. Девід Марк Swift: розробка додатків в середовищі Xcode для iPhone, iPad з використанням iOS SDK / Джек Наттінг, Кім Топлі, Фредрік Т. Опсон, Джефф Ламарш.

5. Дон Норма Дизайн звичних речей., 1988.

6. Джонатан Здзіарскі iPhone SDK. Розробка додатків., 2013., – (БХВ Петербуг)

7. Люк Ворблевскі Спочатку мобільні!, 2012., – (A BOOK APART)

8. Ревич Юрій Цікава електроніка., 2015. – (БХВ Петербург).

9. Ресс Унгер UX-дизайн. Практичний посібник по проектування досвіду взаємодії / Керолайн Чендлер., 2011. – (Профессионально).

10. Роберт Мартін Чистий код. Створення, аналіз та рефакторінг., 2008. – (Питер).

11. Теро Кервінен Робимо сенсори. Проекти сенсорних пристроїв на базі Arduino і Raspberry Pi / Кіммо Карвінен, Вілле Валтокарі., 2015. – (Вильямс).

12. Стів Круг Не змушуйте мене думати., 2000.

13. Уіллі Соммер Програмування мікроконтролерних плат Arduino., 2012. – (БХВ Петербург).

14. Франчіз Переа Основи Ардуіно., 2015. – (Pact Publishing).

15. Apple Мова програмування Swift., 2014.