МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра Комп'ютерних систем та мереж

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
комп'ютерних систем та мереж

_____(Жуков І.А.)

« ____ » _____ 2021 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: Апаратно-програмний комплекс контролю якості повітря

Виконавець: Волощук С.Ю.

Керівник: Проценко М.М.

Нормоконтролер: Журавель С.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних систем та мереж
Напрям (спеціальність) 123 "Комп'ютерна інженерія"
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних систем та мереж

_____(Жуков І.А.)

« ____ » _____ 2021 р.

З А В Д А Н Н Я
на виконання дипломного проєкту

Волощуку Сергію Юрійовичу
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема проєкту (роботи): Апаратно-програмний комплекс контролю якості повітря
затверджена наказом ректора від "26" квітня 2021 року № 648/ст.
2.Термін виконання проєкту (роботи): з 24.05.2021 до 20.06.2021
3.Вихідні дані до проєкту (роботи): спроєктований апаратно-програмний комплекс контролю якості повітря

4.Зміст пояснювальної записки (перелік питань, що підлягають розробці):
Розгляд та аналіз апаратно-програмних комплексів контролю якості повітря.
Розробка апаратного та програмного забезпечення. Створення прототипу.
Визначення індексу якості повітря.

5.Перелік обов'язкового графічного матеріалу:
Презентація *PowerPoint*

6. Календарний план

№ пор.	Завдання	Термін виконання	Відмітка про ввиконання
1.	Ознайомитись з тематикою дипломних проєктів. Вибір теми.	24.05.21 - 26.05.21	
2.	Узгодити технічного завдання.	27.05.21- 01.05.21	
3.	Обробити літературні матеріали за темою дипломного проєкту	02.05.21- 05.05.21	
4.	Проаналізувати апаратно-програмні комплекси	06.05.21- 08.05.21	
5.	Виділити переваги та недоліки апаратно-програмних комплексів.	09.05.21- 11.05.21	
6.	Розглянути структуру апаратно-програмного комплексу	12.05.21- 14.05.21	
7.	Виділити датчики для аналізу якості повітря	15.05.21- 16.05.21	
8.	Розробити схему підключення інтерфейсів до мікроконтролера.	17.05.21- 19.05.21	
9.	Розробити програмне забезпечення комплексу	20.05.21- 01.06.21	
10.	Оформити пояснювальну записку.	02.06.21- 11.06.21	
11.	Підготувати ілюстративні матеріали.	12.06.21- 13.06.21	

7. Дата отримання завдання «24 » травня 2021 р.

Керівник дипломного проєкту Проценко М.М.
(підпис)

Завдання прийняв до виконання Волощук С.Ю.
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту “Апаратно-програмний комплекс контролю якості повітря”: 66 сторінок, 32 рисунка, 12 таблиць, 19 літературних джерел.

АПАРАТНО-ПРОГРАМНИЙ КОМПЛЕКС КОНТРОЛЮ ЯКОСТІ ПОВІТРЯ.

Мета дипломного проєкту – розробка технічного пристрою для контролю якості повітря.

Завдання дипломного проєктування – розробити апаратно-програмний комплекс контролю для вимірювання якості повітря

Об'єкт проєктування – це апаратно-програмний пристрій потрібного для вимірювання якості повітря.

Практична значимість проєкту – полягає у створені технічного пристрою широкого застосування для контролю якості повітря.

ЗМІСТ

Кафедра КСМ				*НАУ 21 10 52 000 ПЗ*			
Виконав	*Волощук С.Ю.*			*Апаратно-програмний комплекс контролю якості повітря*	***Літера***	***Аркуш***	***Аркушів***
Керівник	*Проценко М.М.*					*5*	*66*
Консульт.					*123 КС-431Б*		
Норм. контр.	*Журавель С.В.*						
Зав. Каф.	*Жуков І.А.*						

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

UART – *(Universal Asynchronous Receiver/Transmitter)* універсальний асинхронний приймач/передавач

SPI – *(Serial Peripheral InterFace)* послідовний периферійний інтерфейс

I2C – послідовна асимметрична шина для зв'язку між інтегральними схемами всередині електронних приборів

USB – *(Universal Serial Bus)* універсальна послідовна шина, призначена для з'єднання комп'ютерів і периферійних пристроїв.

SQL – *Structured Query Language* (структурована мова запитів)

TCP/IP – *Transmission Control Protocol/Internet Protocol* (протокол управління передачею/межмережевий протокол)

URL – *Universal Resource Locator* (адреса сторінки в Інтернеті)

БД – база даних

ВСТУП

Забруднення повітря набуло дуже швидких темпів у *XX* ст, адже кількість автомобілів та заводів які забруднюють повітря швидко зростало. Різні хімічні речовини викидаються в повітря як з природних, так і з техногенних (антропогенних) джерел. Кількість може коливатися від сотень до мільйонів тон щороку. Природне забруднення повітря відбувається внаслідок різноманітних біотичних та абіотичних джерел, таких як рослини, радіологічне розкладання, лісові пожежі, вулкани та інші геотермальні джерела, а також викиди із суші та води. Це призводить до концентрації природного фону, яка змінюється залежно від місцевих джерел або конкретних погодних умов.

Антропогенне забруднення повітря існує принаймні з тих пір, як люди навчились користуватися вогнем, але воно швидко зростало з початку індустріалізації. Збільшення забруднення атмосферного повітря внаслідок все більшого використання викопних джерел енергії та зростання виробництва та використання хімічних речовин супроводжувалося посиленням обізнаності та занепокоєння громадськості щодо його шкідливого впливу на здоров'я та навколишнє середовище. Це вимагало розробку апаратного-програмного комплексу для контролю якості повітря у реальному часі.

Контроль якості повітря дуже важливий в наш час, адже від цього залежить здоров’я людини. Комплекс за допомогою індексу якості повітря може попередити вразливих людей, які страждають серцево-судинними або респіраторними хворобами. Також може рекомендувати використання масок, щоб уникнути потраплянню маленьких частинок у легені.

Мета і завдання проєктування – розробка апаратно-програмного комплексу контролю якості повітря, що дозволить в реальному часі отримувати оцінку якості повітря.

Практична значимість проєкту – результати дипломного проєктування рекомендується використовувати під час створення нових комплексів для контролю якості повітря. На даний час є прототип який створенний на основі результату дипломного проєктування.

РОЗДІЛ 1
РОЗГЛЯД АПАРАТНО-ПРОГРАМНИХ КОМПЛЕКСІВ КОНТРОЛЮ ЯКОСТІ ПОВІТРЯ

1.1. Комплекс *Corentium Plus*

Corentium Plus із кількома ультрасучасними датчиками - це монітор радону, який реєструватиме все, що потрібно для перегляду коливань концентрації радону по годинах. Пристрій постачається зручним та сумісним з *Excel* програмним забезпеченням, що дозволяє завантажувати дані на ПК. Детальні технічні характеристики знаходяться в таблиці 1.1.

Рис.1.1. Комплекс *Corentium Plus*

Таблиця 1.1

Технічні характеристики комплексу *Corentium Plus*

Відбір зразків радону:	Пасивна дифузійна камера
Метод виявлення:	Альфа-спектрометрія
Детектор:	1 фотодіод кремнію
Постійна часу дифузії:	25 хв
Діапазон вимірювання:	0 - 50000 Бк / м3
Частота вибірки:	1 година
Операційне середовище:	• від 4 ° C до 40 ° C • Від 5% до 85% відносної вологості без конденсації • від 50 до 110 кПа
Блок живлення:	3 лужні батареї типу *AAA* > 18 місяців часу автономної роботи

Позитивні сторони цього комплексу:

- Мобільність
- Автономна робота від батареї
- Діпазон вимірюваня

Негативні сторони:

- Вимірювання лише радону
- Частота вибірки 1 година
- Не має централізованої точки збору даних

1.2. Комплекс *EG EGVOC*

EG EGVOC цей монітор якості повітря в приміщенні точно оцінює безліч показників якості повітря, включаючи концентрації формальдегіду (*HCHO*), загальних природних та синтетичних летких органічних сполук, що викликають найбільший запах (*TVOC*), і дрібнодисперсних пилоподібних речовин розміром <2,5 мкм (*PM2,5*) які проникають глибоко в дихальні шляхи, досягаючи легеневих

альвеол і можуть погіршити астму. Детальні технічні характеристики знаходяться в таблиці 1.2.

Рис.1.2. Комплекс *EG EGVOC*

Таблиця 1.2

Технічні характеристики комплексу *EG EGVOC*

Назва	**Характеристики**
1	2
Режим відображення:	Рідкокристалічний дисплей високої чіткості (РК) Роздільна здатність 240 x 320
Розмір приладу:	150 x 70,8 x 43,6 мм
Вага приладу:	188 г.
Ємність акумулятора:	2200*mAh*
Операційне середовище	
Атмосферний тиск:	86Kpa - 106Kpa

Продовження табл. 1.2

1	2
Діапазон вологості:	20% - 85%
Робоча температура:	-10°C - 45°C
Формальдегід або *HCHO*	
Тестовий предмет:	атмосферний *HCHO*
Діапазон вимірювання:	0,000-1,999 мг / м3
Метод відбору проб:	дифузія
Одиниця концентрації:	мг / м3
Тип датчика:	електрохімічний напівпровідниковий датчик
Загальні леткі органічні сполуки або *TVOC*	
Тестовий предмет:	*TVOC* у повітрі, включаючи потенційно канцерогенні ароматичні сполуки та бензол
Діапазон вимірювання:	0,000 - 9,999 мг / м3
Метод відбору проб:	дифузія
Одиниця концентрації:	мг / м3
Тип датчика:	електрохімічний напівпровідниковий датчик
PM2,5 / PM10	
Діаметр перевірених частинок:	частинка *PM2,5* або <2,5 мікрометра концентрація (*PM10* через розрахунок за *PM2,5*)
Час вибірки:	3 секунди
Тип датчика:	датчик лазерного виявлення
Діапазон виявлення:	0-999 мкг / м3

Прилад має пороговий рівень тривоги за замовчуванням - 0,10 мг / м3.Однак користувач може встановити інший поріг тривоги *HCHO*, натиснувши ліву кнопку на (+) або праву кнопку на (-) поріг тривоги. Сигнал задзвонить, коли рівень *HCHO* перевищений.

Переваги:

Точкова перевірка якості повітря в приміщенні в будь-якому місці або постійний моніторинг до 2 днів. Легкий в переносі тестер якості повітря може допомогти контролювати якість повітря вдома, в машині та в готельних номерах. Акумулятор на 2200 мАг забезпечує портативне живлення, коли зарядка мікро-*USB* постійним струмом 5 В недоступна. Інформація відображається на чіткому 2,8-дюймовому кольоровому РК-дисплеї.

Недоліки:

Комплекс зберігає дані лише два дні, що є мало для статистики. Також дані лише є на пристрої, та не може бути під'єднаний до системи із декількох пристроїв для контролю якості повітря в регіоні.

1.3. Комплекс *Kronos FT-JQ-002*

Мультифункціональний аналізатор повітря *Kronos FT-JQ-002* - це портативний аналізатор рівня формальдегіду і толуену (*HCHO*), всіх летючих органічних сполук (*TVOC*) і твердих частинок (*PM2.5*). Призначений для аналізу як в закритому приміщенні так і на відкритому просторі. Детальні технічні характеристики знаходяться в таблиці 1.3.

Рис.1.3. Комплекс *Kronos FT-JQ-002*

Таблиця 1.3

Технічні характеристики комплексу *Kronos FT-JQ-002*

Бренд:	*Kronos*
Модель:	*FT-JQ-002*
PM2.5 Діапазон:	*0 - 999 μg / m3*
HCHO Діапазон:	*0 ~ 1.999mg / m3*
PM10 Діапазон:	*0 ~ 999 μg / m3*
PM1.0 Діапазон:	*0 ~ 999 μg / m3*
Відхилення:	0,001
Вага:	185 г
Розмір:	15 см x 6,5 см
Ємність акумулятора:	2200*mAh*

Переваги:

Портитивність дає змогу точкової перевірки якості повітря в приміщенні в будь-якому місці. Легкий в переносі може допомогти контролювати якість повітря вдома, в машині та в готельних номерах. Акумулятор на 2200 мАг забезпечує портативне живлення. Призначений для аналізу як в закритому приміщенні так і на відкритому просторі.

Недоліки:

Комплекс не зберігає дані для статистики. Не має можливості зміни порогу перевищення забруднення при якому включається звуковий сигнал. Спектр вимірювальних даних малий.

1.4. Комплекс *PCE-RCM 12*

Комплекс *PCE-RCM12* служить для орієнтовного вивчення параметрів навколишнього середовища. Комплекс дозволяє вимірювати дрібний пил, температуру, відносну вологість і концентрацію формальдегіду і вуглекислого газу. Яскравий дисплей надає інформацію про вже згаданих параметрах. Комплекс може працювати від вбудованого акумулятора безперервно до 5 годин. Для постійного вимірювання аналізатор можна підключити до мережі через *USB*-інтерфейс. Підрахунок частинок застосовується для пилу декількох розмірів: 2,5 мкм і 10 мкм.

Цей пристрій є детектором якості повітря, що включає декілька вимірювачів, інтелектуальну індукцію та хмарне зберігання даних. Може виявляти *PM2,5, PM10, C02, HCHO, TVOC,* температуру та вологість. Цей пристрій використовує метод випробування на вагу лазерного розсіювання з професійною конструкцією конструкції, професійним вентилятором та оригінальними частинками. Може чутливо і точно виявляти та контролювати *PM2,5* та *PM10*, допомагати вам знати якість повітря у приміщеннях та вживати заходів, щоб зробити повітря чистим. Використовує професійний датчик *CO2*, може нагадувати вам про відкриття вікон, коли концентрація *CO2* у приміщенні занадто висока.

HCHO може виявляти формальдегід, а *TVOC* - виявляти шкідливий газ у приміщенні, допомагати вам знати забруднення навколишнього середовища. стабільний і час роботи максимально довгий. Модуль виявлення шуму обладнаний.

Ви можете пробудити пристрій для початку вимірювання, створюючи невеликий шум, коли пристрій спить. Детальні технічні характеристики знаходяться в таблиці 1.4.

Рис.1.4. Комплекс *PCE-RCM 12*

Таблиця 1.4

Технічні характеристики комплексу *PCE-RCM 12*

Назва	**Характеристики**
1	2
Вимірювання часток	
Канали частинок:	*PM2,5 / PM10*
Діапазон масової концентрації:	0-2000 мкг /м^3
Роздільна здатність:	1 мкг/м^3
Вимірювання *CO_2*	
Діапазон:	0-9999 *ppm*
Точність:	± 5% ± 75 *ppm*
Роздільна здатність:	1 *ppm*
Вимірювання *HCHO*	

Продовження табл. 1.4

1	2
Діапазон:	0,00-5,00 мг/м^3
Точність:	± 5%
Роздільна здатність:	0,01 мг/м ^{3}
Вимірювання *TVOC*	
Діапазон:	0,00-9,99 мг/м^3
Точність:	± 5%
Роздільна здатність:	0,01 мг/м ^{3}
Вимірювання температури та вологості	
Діапазон температур: Точність температури ± 2 "C Дозвіл температури 0,1 "C Діапазон вологості 0-100% вологості Точність вологості ± 3,5% вологості (20-80% вологості)	-20-70 °*C* (-4-158° *F*)
Точність температури:	± 2 °*C*
Роздільна здатність температури:	0,1 °*C*
Діапазон вологості:	0-100% вологості
Точність вологості:	± 3,5% вологості (20-80% вологості)
Інші	
Ємність акумулятора:	2200*mAh*
Дисплей:	3-дюймовий *TFT*-РК-дисплей, 240 * 400пікселів
Зберігання даних:	5000 груп даних про вибірку
Розмір:	85 * 75 * 155 мм
Вага:	360г

Переваги:

Вимірювання температури та вологості, невеликий розмір та вага. Має можливісь зберігати 5000 груп даних про вибірку.

Недоліки:

Комплекс зберігає дані лише на своєму приладі, що є мало для статистики. Також дані лише є на пристрої, та не може бути під'єднаний до системи із декількох пристроїв для контролю якості повітря в регіоні, та оцінки забрудненості повітря.

1.5. Комплекс *SaveEcoSensor*

SaveEcoSensor - це станція моніторингу якості повітря яка дозволяє вимірювати вміст пилу фракцій 2.5 та 10 мікронів в повітрі (так звані *PM 2.5* і *PM 10*). Інтегрований сенсор температури-вологості-тиску дозволяє автоматично коригувати отриману інформацію в залежності від погодних умов, а наявність модулю підігріву дозволяє мінімізувати вплив під час туману, опадів та при від'ємних температурах. Складові частини приладу *SaveEcoSensor* наступні:

- Пиломір *SDS011*.
- Контролер *Wemos D1 mini V2 Pro.*
- Сенсор температури, вологості та тиску *BME280.*
- Пластиковий корпус.
- Блок живлення.
- Автоматизована камера підігріву повітря.

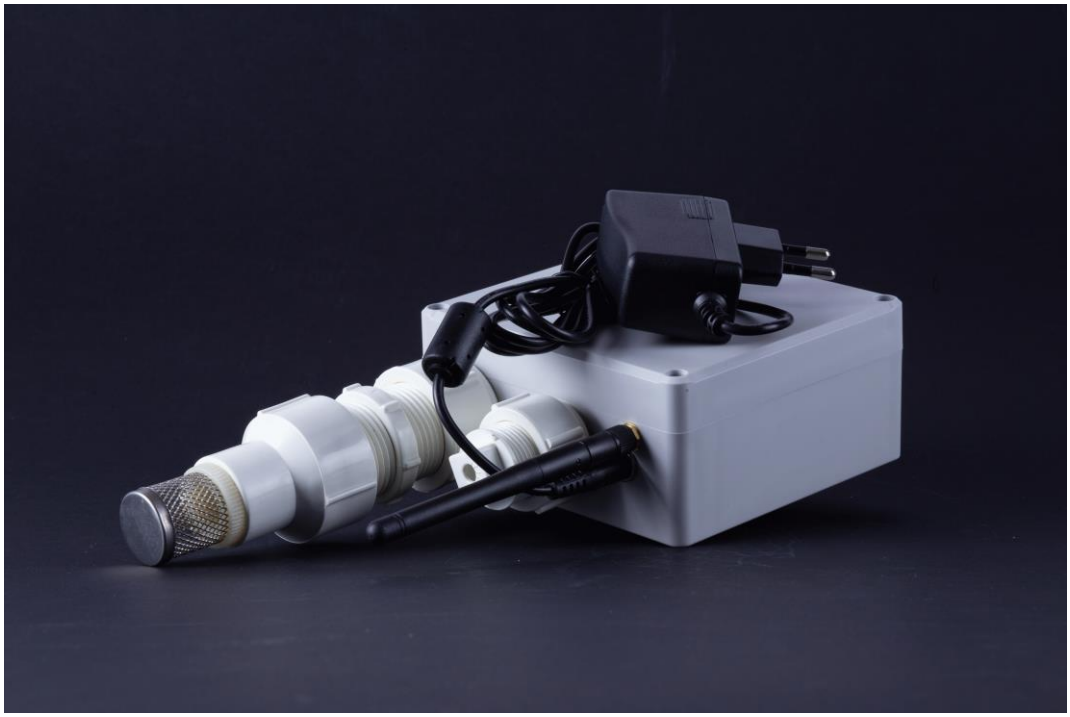

Рис.1.5. Комплекс *SaveEcoSensor*

Прилад, підключений до мережі, проводить вимірювання кожні 145 секунд та відправляє дані напряму до наступних онлайн-ресурсів:

1. Система *SaveEcoBot*, яка складається з інтерактивної мапи та чат-бота.
2. *aqicn.org* – глобальна карта забруднення повітря, яка охоплює весь світ.
3. *OpenSenseMap.org* – відкрита карта, де кожен бажаючий має можливість додати сенсор та публікувати дані. Має зручний інтерфейс, *API*, можливість візуалізації даних за допомогою інтерполяції та навіть візуального перегляду даних у часі – анімація того, як змінювались показники на карті. Наприклад, коли покриття міста приладами буде повноцінним, можливо буде відстежувати, як переміщується хмара пилу.
4. Сервер *LuFtdaten.inFo* .
5. Сервер *Madavi.de* – сервер зберігання даних, розроблений німцями, для того щоб отримувати дані у форматі *CSV* та дивитися графіки з даними сенсорів приладів.
6. Онлайн карта розробників з Івано-Франківську *eco-city.org.ua*.
7. *air-pollution.ml* – мапа забруднення повітря України.

Переваги:

Вимірювання температури та вологості, невеликий розмір та вага.Підігрів повітря, що дає змогу автоматично коригувати отриману інформацію в залежності від погодних умов. Відправка даних напряму на велику кількість ресурсів, які забезпечують інформування користувачів в реальному часі.

Недоліки:

Комплекс вимірює дуже малий спектр газів, які забруднюють повітря. Комплекс не має буфера даних на приладі.

1.6. Комплекс *Outdoor Air Quality Test Kit* (*Pro*)

Комплекс *Outdoor Air Quality Test Kit (Pro)* призначений для консультантів та фахівців з якості повітря, які хочуть отримати повний набір інструментів для вимірювання ряду загальних забруднюючих речовин у зовнішньому зовнішньому повітрі.

Outdoor Air Quality Test Kit (Pro) оснащений перевіреним портативним монітором серії 500 компанії *Aeroqual*, який можна використовувати для вимірювання ряду забруднюючих речовин, просто замінюючи головки датчиків на забруднювач, який ви хочете контролювати. У комплект входять такі датчики: датчик твердих часток *(PM2,5 / PM10)*, чотири датчики газу, що забруднює газ *(NO2, O3, CO, VOC)*, а також комбінований датчик температури та відносної вологості.

Детальні технічні характеристики знаходяться в Таблиці 1.5.

Рис.1.6. Комплекс *Outdoor Air Quality Test Kit (Pro)*

Таблиця 1.5

Технічні характеристики комплексу *Outdoor Air Quality Test Kit (Pro)*

Назва	**Характеристики**
1	2
Головка датчика твердих частинок *PM10* / *PM2.5*	
Діапазон:	від 0,000 до 1000 мг / м3
Параметри вимірювання :	*PM2.5* та *PM10*
Тип датчика:	Лазерний лічильник частинок
Мінімальна межа виявлення :	0,001 мг / м3
Точність заводського калібрування:	± (0,002 мг / м3 + 15% показань)
Роздільна здатність:	0,001 мг / 3
Час відгуку:	5 секунд
Температура:	від 0 до 40 ° С
Відносна вологість:	від 0 до 90% без конденсації
Датчик озону	

Продовження табл. 1.5

1	2
Діапазон :	0-0,15*ppm*
Тип датчика:	*GSS*
Мінімальна межа виявлення :	0,001 *ppm*
Точність заводського калібрування:	<± 0,005 *ppm*
Роздільна здатність:	0,001 *ppm*
Час відгуку	60 секунд
Температура	від 0 до 40 ° С
Відносна вологість	від 10 до 90%
Датчик діоксиду азоту	
Діапазон :	0-1*ppm*
Тип датчика:	*GSE*
Мінімальна межа виявлення :	0,005 *ppm*
Точність заводського калібрування:	<±0.02 *ppm* 0-0.2 *ppm* <±10% 0.2-1 *ppm*
Роздільна здатність:	0,001 *ppm*
Час відгуку	30 секунд
Температура	від 0 до 40 ° С
Відносна вологість	від 15 до 90%
Датчик чадного газу	
Діапазон :	0-25*ppm*
Тип датчика:	*GSE*
Мінімальна межа виявлення :	0,05 *ppm*
Точність заводського калібрування:	<±0.5 *ppm* 0-5*ppm* <±10% 5-25*ppm*
Роздільна здатність:	0,01 *ppm*
Час відгуку	60 секунд
Температура	від 0 до 40 ° С

Продовження табл. 1.5

1	2
Відносна вологість	від 15 до 90%
Датчик летких органічних речовин	
Діапазон :	0-25*ppm*
Тип датчика:	*GSS*
Мінімальна межа виявлення :	0,1 *ppm*
Точність заводського калібрування:	<±0.1 *ppm* + 10%
Роздільна здатність:	0,1 *ppm*
Час відгуку	60 секунд
Температура	від 0 до 40 ° C
Відносна вологість	від 19 до 90%

Переваги:

- Повний комплект контролю якості повітря на відкритому повітрі для вимірювання критеріїв забруднювачів повітря та летких органічних речовин
- Монітор серії 500 із вбудованим реєстратором даних у режимі реального часу
- Активні головки датчиків для вибірки вентилятора (*PM2,5 / PM10, NO2, O3, CO, TVOC*) забезпечують високу точність вимірювань
- Технологія літієвих акумуляторів тривалого терміну служби

Недоліки:

Комплекс зберігає дані лише на своєму приладі, що є мало для статистики. Також дані лише є на пристрої, та не може бути під'єднаний до системи із декількох пристроїв для контролю якості повітря в регіоні, та оцінки забрудненості повітря.

Висновки до розділу

Серед розглянутих комплексів, найбільш вдалою для створення апаратно-програмного комплексу контролю якості повітря використання модулі датчиків: цифровий модуль датчика газу для вуглецю (*CO*), азоту (NO_2), сірки(SO_2) , озону(O_3), датчик якості повітря *PM2.5* пилу, датчик *eCO2 (*еквівалентний *CO2)* та

TVOC (загальна кількість летких органічних сполук*),* комплексний датчик *HCHO* - этану (*CH3*), бензену (*C6H6*), формальдегіду *(H2CO).* Тоді комплекс зможе надавати максимальний спектр даних і давати коректну оцінку якості повітря. Повинен мати фнутрішню пам’ять для зберігання даних, або відправляти дані на віддалений сервер.

Створення централізоване сховище даних, за допомогою якого можливу оброблювати інформацію та відображення інформації про якість повітря на сайті. Це допоможе вчасно надавати інформацію про якість, та аналізувати причини забруднення.

Використовувати акумулятор в комплексі не доцільно, так як комплекс є стаціонарним і потребує підключення до мережі інтернет для збереження даних та обробки даних.

РОЗДІЛ 2

СТРУКТУРА АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ

2.1. Структура апаратно-програмного комплексу

Апаратно-програмний комплекс можна розділити на дві основні частини: апаратна та програмна. Апаратна частина складається з трьох частин: камера для забору повітря, основна частина з мікроконтролером, та одноплатний міні-комп'ютер *Orange Pi.*

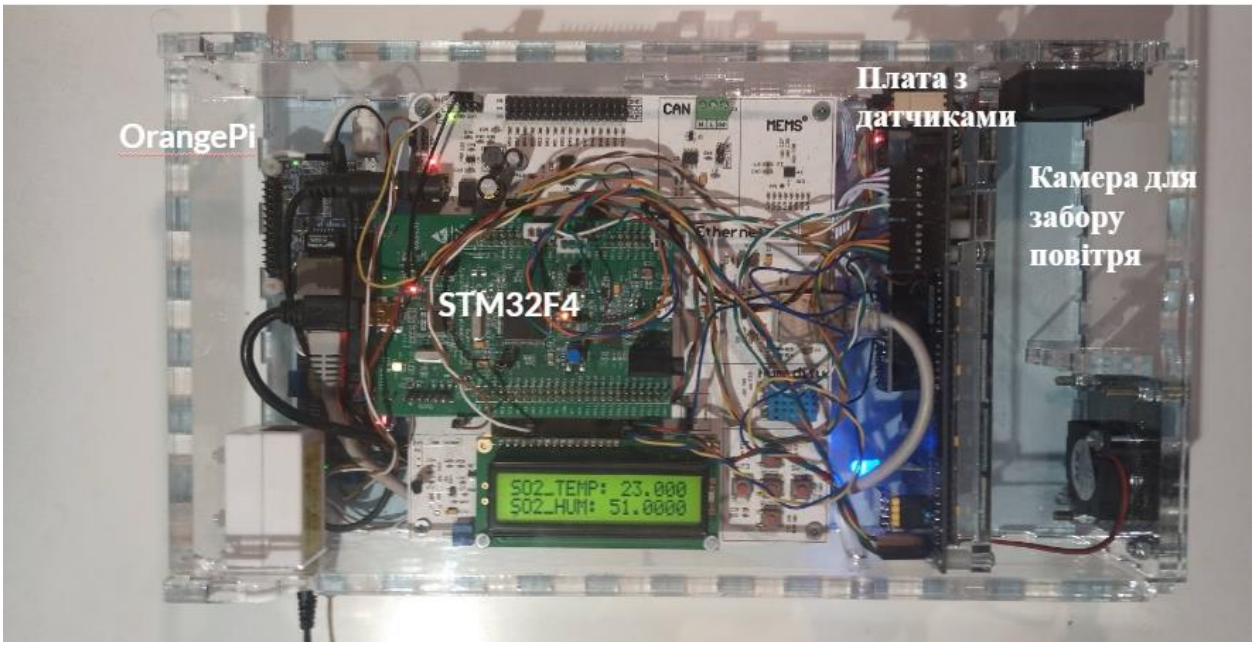

Рис.2.1. Апаратно-програмний комплекс

2.2. Камера для забору повітря

Камера для забору повітря складається з двох вентиляторів якими керує мікроконтроллер, які кожного разу завантажують в камеру свіже повітря та модулю на якому знаходяться датчики які вимірюють якість повітря.

Рис.2.2. Модуль з датчиками

Модуль складається з друкованої плати на якій розведені доріжки. На друкованій платі розпаяні модулі датчиків: цифровий модуль датчика газу для окису вуглецю (*CO*) *DGS-CO*(технічні характеристики наведені в таблиці 2.1) , азоту (NO_2) *DGS-NO2*(технічні характеристики наведені в таблиці 2.2), сірки(SO_2) *DGS-SO2*(технічні характеристики наведені в таблиці 2.3), озону(O_3) *DGS-O3*(технічні характеристики наведені в таблиці 2.4), датчик якості повітря *PM2.5* пилу *SDS011*, датчик *eCO2 (*еквівалентний *CO2)* та *TVOC* (загальна кількість летких органічних сполук*) CCS811*, комплексний датчик *HCHO* - этану (*CH3*), бензену (*C6H6*), формальдегіду *(H2CO) SEN0231*, модуль погоди *BME280*. Також на платі знаходиться мультиплексор *CD74HC4067*, та модуль *Wi-Fi ESP8266*.

Таблиця 2.1

Технічні характеристики *DGS-CO*

Назва	**Характеристики**
1	2
Діапазон вимірювання	від 0 *до* 1000 *ppm*
Роздільна здатність	0,1 *ppm*
Нульова точність	± 1 *ppm*
Точність вимірювання	15%
Повторюваність вимірювання	<± 3% від показань або 2 *ppm*, залежно від того, що більше
T90 Час відгуку (крок 100 *ppm*)	<30 секунд (типово 15 секунд)

Продовження табл. 2.1

1	2
Споживання енергії	1 мВт протягом 1 хв 12 мВт для безперервної вибірки з інтервалами 5, 10 30, 60 секунд
Очікуваний експлуатаційний ресурс	> 5 років (10 років при 25 ± 10С; 60 ± 30% вологості)
Діапазон робочої вологості	Від 15 до 95% (від 0 до 100% без конденсації з перервами)
Механічні розміри	1,75 х 0,82 х 0,35 дюйма (44,5 х 20,8 х 8,9 мм)
Вага	<2 унцій

Таблиця 2.2

Технічні характеристики *DGS-NO2*

Назва	**Характеристики**
1	2
Діапазон вимірювання	*від* 0 *до* 5 *ppm*
Роздільна здатність	20 *ppb*
Точність вимірювання	15%
Повторюваність вимірювання	<± 3% від показань
Т90 Час відгуку (крок 100 *ppm*)	<30 секунд
Споживання енергії	100 мкВт в режимі очікування 14 мВт в режимі вимірювання
Очікуваний експлуатаційний ресурс	> 5 років (10 років при 25 ± 10*C*; 60 ± 30% вологості)
Діапазон робочих температур	-20 до 40 ° *C* (-30 до 55 ° *C* з перервами)
Діапазон робочої вологості	Від 15 до 95% (від 0 до 100% без конденсації з перервами)

Продовження табл. 2.2

1	2
Механічні розміри	1,75 х 0,82 х 0,35 дюйма (44,5 х 20,8 х 8,9 мм)
Вага	<2 унцій

Таблиця 2.3

Технічні характеристики *DGS-SO2*

Діапазон вимірювання	*від* 0 *до* 20 *ppm*
Роздільна здатність	50 *ppb*
Точність вимірювання	15%
Повторюваність вимірювання	<± 3% від показань
T90 Час відгуку (крок 100 *ppm*)	<30 секунд
Споживання енергії	1 мВт протягом 1 хв 12 мВт для безперервної вибірки з інтервалами 5, 10 30, 60 секунд
Очікуваний експлуатаційний ресурс	> 5 років (10 років при 25 ± 10С; 60 ± 30% вологості)
Діапазон робочих температур	-20 до 40 ° С (-30 до 55 ° С з перервами)
Діапазон робочої вологості	Від 15 до 95% (від 0 до 100% без конденсації з перервами)
Механічні розміри	1,75 х 0,82 х 0,35 дюйма (44,5 х 20,8 х 8,9 мм)
Вага	<2 унцій

Таблиця 2.4

Технічні характеристики *DGS-O*3

Назва	**Характеристики**
Діапазон вимірювання	*від* 0 *до* 5 *ppm*
Роздільна здатність	20 *ppb*
Точність вимірювання	15%
Повторюваність вимірювання	<± 3% від показань
T90 Час відгуку (крок 100 *ppm*)	<30 секунд
Споживання енергії	100 мкВт в режимі очікування 14 мВт в режимі вимірювання
Очікуваний експлуатаційний ресурс	> 5 років (10 років при 25 ± 10*C*; 60 ± 30% вологості)
Діапазон робочих температур	-20 до 40 ° C (-30 до 55 ° *C* з перервами)
Діапазон робочої вологості	Від 15 до 95% (від 0 до 100% без конденсації з перервами)
Механічні розміри	1,75 x 0,82 x 0,35 дюйма (44,5 x 20,8 x 8,9 мм)
Вага	<2 унцій

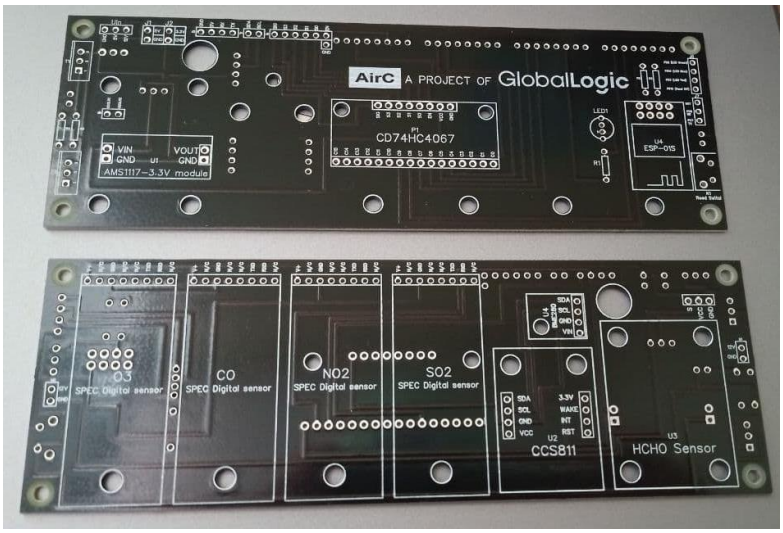

Рис.2.3. Друкована плата для датчиків

Мультиплексор *CD74HC4067* (Рис.2.4) потрібен для підключення датчиків *DGS-CO, DGS-NO2, DGS-SO2, DGS-O3, SEN0231* та *SDS011*. Мультиплексор був використаний тому що кількість інтерфейсів *UART* у мікроконтролера *STM32F4* було не достатньо.

UART у перекладі з англійського як “Універсальний асинхронний прийомопередатчик”, він є одним із прмітивних інтерфейсів для підключення перефирій. Кожен біт кожного байта передається в рівний відведений проміжок часу (фактично, тайм-слот). Стандартним розміром даних в пакеті є 8 байт, але крім даних кожен пакет несе і службову інформацію, а саме: стартовий біт, (Обов'язковий), стоповий біт (Також обов'язковий, можливе використання 1, 1.5, 2 степових бітів), біт парності (Необов'язковий). Так як інтерфейс асинхронний, то більшої значущості має швидкість передачі даних - і у приймача, і у передавача вона повинна бути однаковою.

CD74HC4067 і *CD74HCT4067* - це цифрові керовані аналогові вимикачі, які використовують кремній-затворну технологію *CMOS* для досягнення робочих швидкостей, подібних до *LSTTL*, з низьким енергоспоживанням стандартних інтегрованих схем *CMOS*. перемикачі, таким чином дозволяючи будь-якому аналоговому входу використовуватись як вихідний сигнал та навпаки. Вимикачі мають низький опір "увімкнення" та низькі "вимкнення" витоків. Крім того, ці пристрої мають ввімкнене управління, яке при високому рівні вільного режиму перемикається у свій вимкнений стан

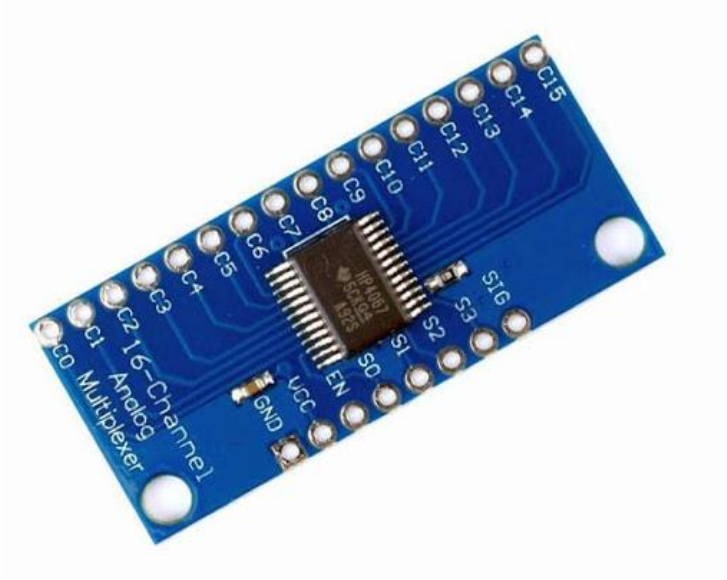

Рис.2.4. Мультиплексор *CD74HC4067*

Модуль *Wi-Fi ESP8266* (Рис.2.5), застосовується для конфігурації комплексу, та передачі даних на віддалений сервер. В таблиці 2.5 наведені технічні характеристики.

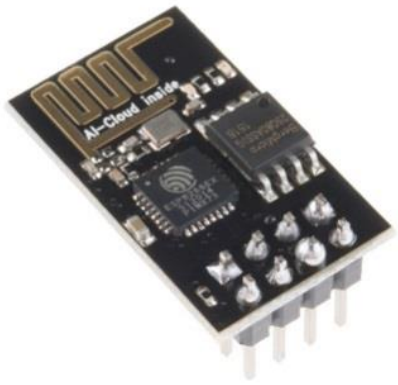

Рис.2.5. Модуль *Wi-Fi ESP8266*

Таблиця 2.5

Технічні характеристики *Wi-Fi ESP8266*

Категорії	Предмети	Параметри
1	2	3
Wi-Fi	Сертифікація	*Wi-Fi Alliance*
	Протоколи	*802.11 b/g/n (HT20)*
	Діапазон частот	*2.4 GHz ~ 2.5 GHz (2400 MHz ~ 2483.5 MHz)*
Апаратне забезпечення	ЦП	32-розрядний процесор *Tensilica L106*
	Периферійний інтерфейс	*UART/SDIO/SPI/I2C/I2S*
	Робоча напруга	2,5 В ~ 3,6 В
	Середнє значення робочого струму	80 мА
	Діапазон робочих температур	–40 ° C ~ 125 ° C
	Розмір упаковки	*QFN32*-контактний (5 мм x 5 мм)

Продовження табл. 2.5

1	2	3
Програмне забезпечення	Режим *Wi-Fi*	Станція / *SoFtAP* / *SoFtAP* + Станція
	Безпека	*WPA/WPA2*
	Шифрування	*WEP/TKIP/AES*
	Оновлення мікропрограми	Завантаження *UART* / *OTA* (через мережу)
	Розробка програмного забезпечення	Підтримує розробку хмарних серверів / прошивки та *SDK* для швидкого програмування на мікросхемі
	Мережеві протоколи	*IPv4, TCP/UDP/HTTP*
	Конфігурація користувача	Набір інструкцій *AT*, хмарний сервер, додаток для *Android*

Функціональне призначення камери для забору повітря є аналіз повітря середовища, та відправка даних на мікроконтроллер.

2.3. Основна частина з мікроконтролером

STM32 - сімейство 32-бітних мікроконтролерів виробництва *STMicroelectronics*. Чіпи *STM32* групуються в серії, в рамках кожної з яких використовується один і той же 32-бітове ядро *ARM*, наприклад, *Cortex-M7F*, *Cortex-M4F*, *Cortex-M3*, *Cortex-M0* + або *Cortex-M0*. Кожен мікроконтролер складається з ядра процесора, статичної *RAM*-пам'яті, флеш-пам'яті, відладочного і різних периферійних інтерфейсів. Розроблений для медичних, промислових та споживчих застосувань, де потрібен високий рівень інтеграції та продуктивності, вбудована пам’ять та багатий периферійний набір усередині розміром від 10 х 10 мм.

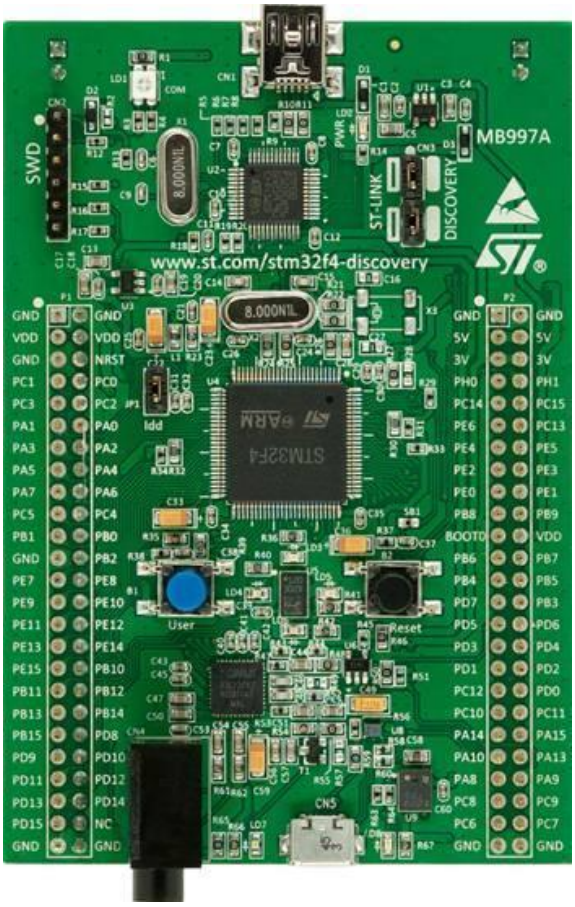

Рис.2.6. Мікроконтролер *STM32F407*

Основна частина складається з мікроконтролера *STM32F417*(Рис.2.6.). *STM32F407* пропонує продуктивність ядра *Cortex* ™ *-M4* (з блоком з плаваючою комою), що працює на частоті 168 МГц.

Продуктивність: При частоті 168 МГц *STM32F407* забезпечує продуктивність 210 *DMIPS* /566 *CoreMark*, що виконується з флеш-пам'яті, із станами 0-очікування за допомогою *ART Accelerator ST*. Інструкції *DSP* та одиниця з плаваючою комою розширюють спектр адресних програм. Енергоефективність: 90-нм процес *ST*, *ART*-прискорювач та динамічне масштабування потужності дозволяють споживати струм у режимі роботи та виконувати з флеш-пам'яті до 238 мкА / МГц на 168 МГц.

Приферія:

Ethernet MAC10/100 з підтримкою *IEEE 1588 v2* та 8–14-бітним паралельним інтерфейсом камери для підключення датчика камери *CMOS*.

2x USB OTG (один із підтримкою *HS*)

Аудіо: виділений аудіо *PLL* та 2 повнодуплексних *I²S*

До 15 інтерфейсів зв'язку (включаючи 6x *USART*, що працюють зі швидкістю до 11,25 Мбіт / с, 3x *SPI*, що працюють зі швидкістю до 45 Мбіт / с, 3x *I²C*, 2x *CAN*, SDIO)

Аналоговий: два 12-бітових ЦАП, три 12-бітові АЦП, що досягають 2,4 *MSPS* або 7,2 *MSPS* в режимі чергування

До 17 таймерів: 16- і 32-бітні, що працюють на частоті до 168 МГц

Легко розширюваний діапазон пам'яті за допомогою гнучкого статичного контролера пам'яті, що підтримує пам’яті *Compact Flash, SRAM, PSRAM, NOR* та *NAND*

Інтеграція: *STM32F417x* забезпечує від 512 Кбайт до 1 Мбайт флеш-пам'яті.

Мікроконтролер викнонує декілька функцій:

• Збір та обробка даних з датчиків

• Управління вентиляторами камери для забору повітря

• Збереження конфігурацій на флеш пам’яті

• Формування пакету даних датчиків та посилання їх через *Ethernet* за допомогою протоколу *TCP*

• Вивід інформації на дисплей

• Управління світодіодами про стан пристрою

Для написання програмного забезпечення використовувалось інтегроване середовище розробки *STM32CubeMX*. *STM32CubeMX* - це графічний інструмент, який дозволяє дуже легко конфігурувати мікроконтролери та мікропроцесори *STM32*, а також генерувати відповідний код ініціалізації C для ядра *Arm® Cortex®-*, через покроковий процес.Перший крок полягає у виборі: мікроконтролера *STMicrolectronics STM32*, мікропроцесора або платформи розробки, яка відповідає необхідному набору периферійних пристроїв, або прикладу, що працює на певній платформі розробки.

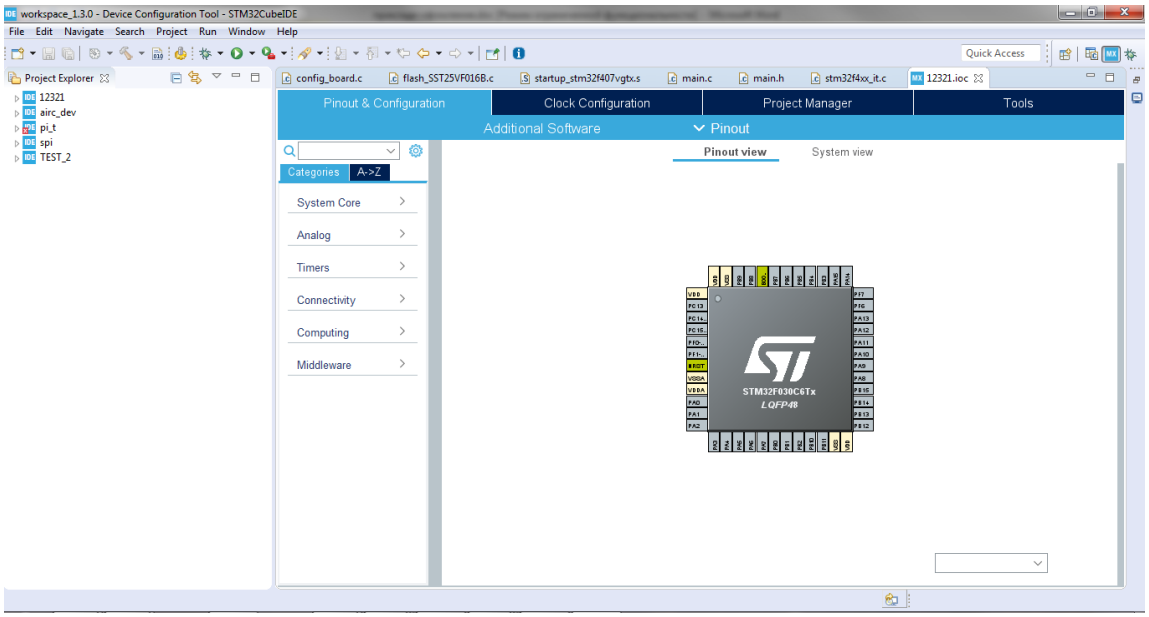

Рис.2.7. Інтерфейс інтегроване середовища розробки *STM32CubeMX*

Для мікропроцесорів другий крок дозволяє налаштувати *GPIO* і налаштування годинника для всієї системи, а також інтерактивно призначити периферію або *Arm® Cortex®-M,*. Спеціальні утиліти, такі як конфігурація та настройка *DDR*, полегшують початок роботи з мікропроцесорами *STM32*. Для ядра *Cortex®-M* конфігурація включає додаткові кроки, які точно відповідають описаним для мікроконтролерів.

Для мікроконтролерів та мікропроцесора *Arm® Cortex®-M* другий крок полягає у налаштуванні кожного необхідного вбудованого програмного забезпечення завдяки вирішувачеві конфлікту між розсипками, помічнику настройки годинника, калькулятору енергоспоживання та утиліті, яка налаштовує периферію (такі як *GPIO* або *USART*) та стеки проміжного програмного забезпечення (наприклад, *USB* або *TCP / IP*).

Стандартні пакети програмного забезпечення та проміжного програмного забезпечення можна розширити завдяки вдосконаленим пакетам розширення *STM32Cube*. *STMicrolectronics* або партнерські пакети *STMicrolectronics* можна завантажити безпосередньо зі спеціального диспетчера пакетів, доступного в *STM32CubeMX*, тоді як інші пакети можна встановити з локального диска.

2.4. Інформаційний дисплей.

STM32F407 керує дисплеєм *WH1602B-NYG-CT*. *WH1602B-NYG-CT* – це рідкокристалічний екран-індикатор з можливості виводу символів розміру 16*x*2.Що дає змогу оперативно отримувати інформацію в реальному часі про якість повітря. Схема підключення дисплея наведенна на рисунку 2.8.

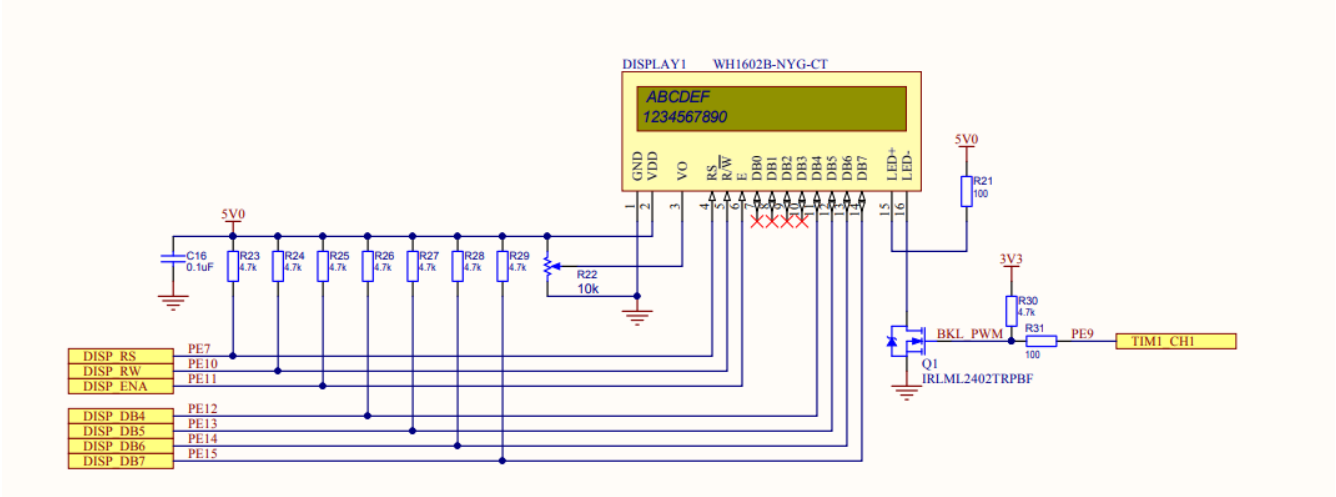

Рис.2.8. Схема підключення дисплея *WH1602B-NYG-CT*

Модуль має вбудований контролер: *ST7066*. Функції контролера контактного інтерфейсу знаходяться в таблиці.2.6.

Таблиця 2.6

Функції контролера контактного інтерфейсу

Пін №	Символ	Опис
1	V_{SS}	Земля
2	V_{DD}	Блок живлення для логіки
3	V_O	Регулювання контрасту
4	*RS*	Сигнал вибору даних / інструкцій
5	*R/W*	Сигнал вибору читання / запис
6	*E*	Увімкнути сигнал
7~14	*DB0~DB7*	Лінія шини даних
15	*A*	Блок живлення для *B / L +*
16	*K*	Блок живлення для *B / L -*

2.5. Датчик температури *LM335Z*

Для визначення температури використовувався датчик температури *LM335Z*. *LM335Z* - це точний датчики температури, який можна легко калібрований. Він працює як 2-термінальний стабілітрон, а напруга пробою прямо пропорційна до абсолютної температури 10 мВ / ° К. Схема має динамічний опір менше 1 Ом і працює в діапазоні струму від Від 450мкА до 5мА без його зміни характеристики. Калібровано при + 25 ° С, має типову похибку менше 1 ° С над а Діапазон температур 100 ° С. Схема підключення датчика наведенна на рисунку 2.9.

Для считування даних з датчика використвується аналого-цифровий перетворювач - пристрій, що перетворює вхідний аналоговий сигнал у дискретний код (цифровий сигнал).

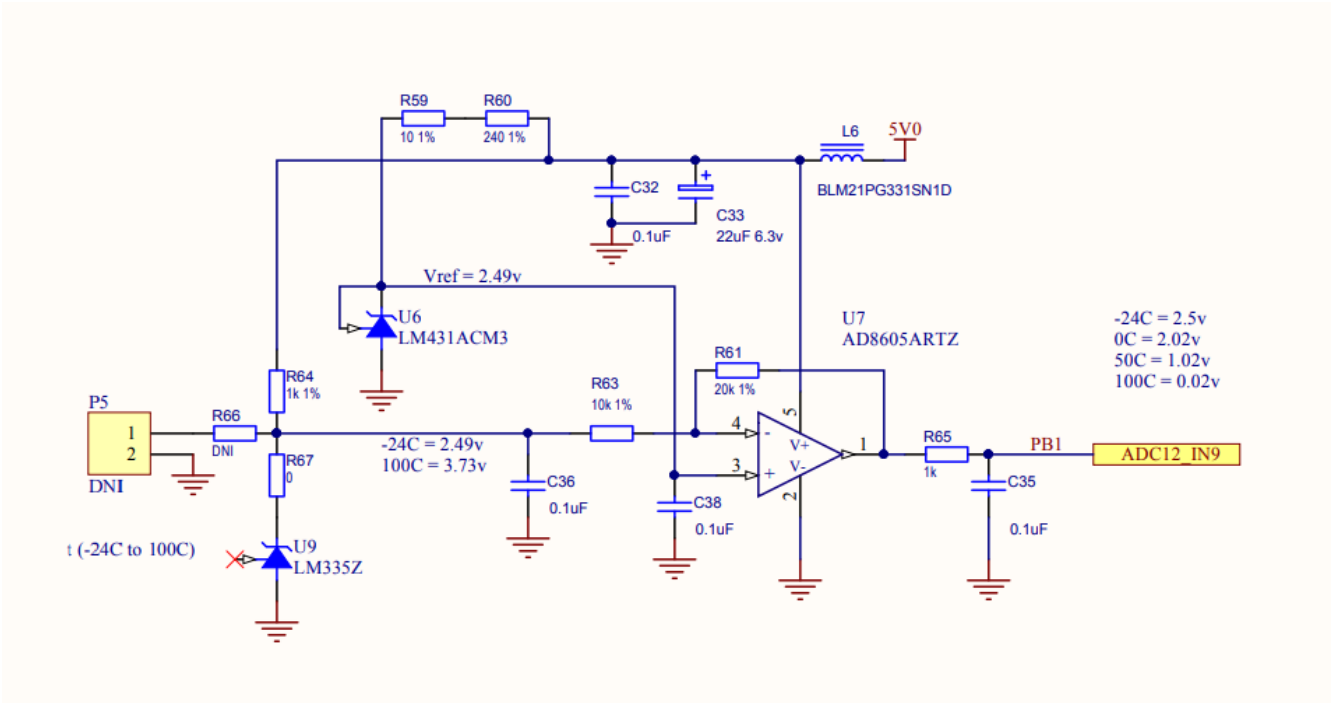

Рис. 2.9. Схема підключення датчика *LM335Z*

2.6. Флеш пам’ять *SST25VF016B*

Флеш пам’ять *SST25VF016B* має чотирипровідний, сумісний з *SPI* інтерфейс, що забезпечує низьку кількість контактів, що займає менше місця на платі і в кінцевому підсумку знижує загальні системні витрати. Пристрої *SST25VF016B* покращені покращеною робочою частотою, що знижує енергоспоживання. Серійні флеш-пам’яті *SPI SST25VF016B SPI* виготовляються за допомогою власної

високопродуктивної технології *CMOS SuperFlash* від *SST*. Конструкція роздільних воріт та тунельний інжектор з товстими оксидами досягають кращої надійності та технологічності порівняно з альтернативними підходами. Схема підключення флеш пам'яті наведенна на рисунку 2.10.

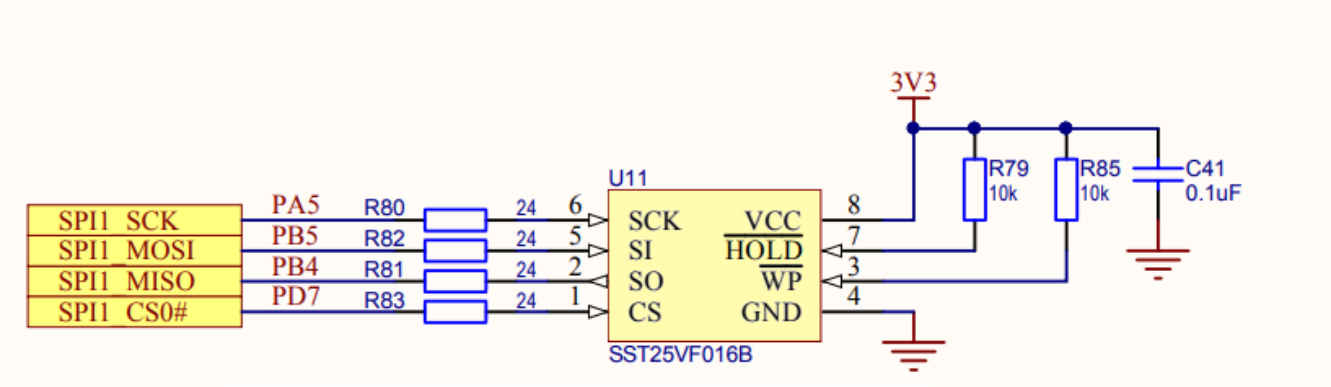

Рис. 2.10. Схема підключення флеш пам'яті *SST25VF016B*

2.7. Одноплатний міні-комп'ютер *Orange Pi*

Orange Pi представлений у вигляді одноплатного міні-комп'ютера на базі чотирьох ядерного процесора *Allwinner H6 ARM Cortex-A53 Quad-core* 1.8*GHz 64 bit*. *Orange Pi* має невеликі розміри, наявність інтерфейсів *HDMI, USB, micro SD, Ethernet Gigabit, IR-port, GPIO* надають перевагу для застосування в різних сферах: «розумний будинок», системи відеонагляду, медіасервери та інших. На ньому встановлена операційна система *Ubuntu*, та запущені сервіси які виконують функції:

1) Прийом данних від мікроконтроллера *STM32F4* (через інтерфейс *Ethernet* за допомогою протоколу *TCP*).
2) Р*еєстрація даних*
3) Приєднує дані до пакету (*json*)
4) Збирає інформацію з усіх пристроїв, працює як сітчастий вузол для збору повідомлення від пристроїв та інфраструктури, а потім відправляє зведені дані в хмару або від хмари до певного пристрою чи вузла.

   В рамках комлексу:

- Збирає повідомлення та буферизує їх
- Перетворює *json* на *MQTT*
- Кодує пакет

- Надсилає пакет на хмару *Google*

2.8. Схема передачі даних апаратно-програмного комплексу

Комплекс використовує різні інтерфейси та протоколи для передачі даних. Для комунікації з датчиками використовуються такі інтерфейси:

- *UART* для підключення датчиків *DGS-CO*, *DGS-NO2*, *DGS-SO2*, *DGS-O3*, *SEN0231* та *SDS011*, *ESP8266EX*
- *I2C* для датчика *CCS811*, *BME280*
- *SPI* для флеш-пам'яті *SST25VF016B*, *Wi-Fi ESP8266*

На Рис 2.11 зображена повна схема всіх інтерфейсів для підключення датчиків та переферій яка потрібна для комлесу.

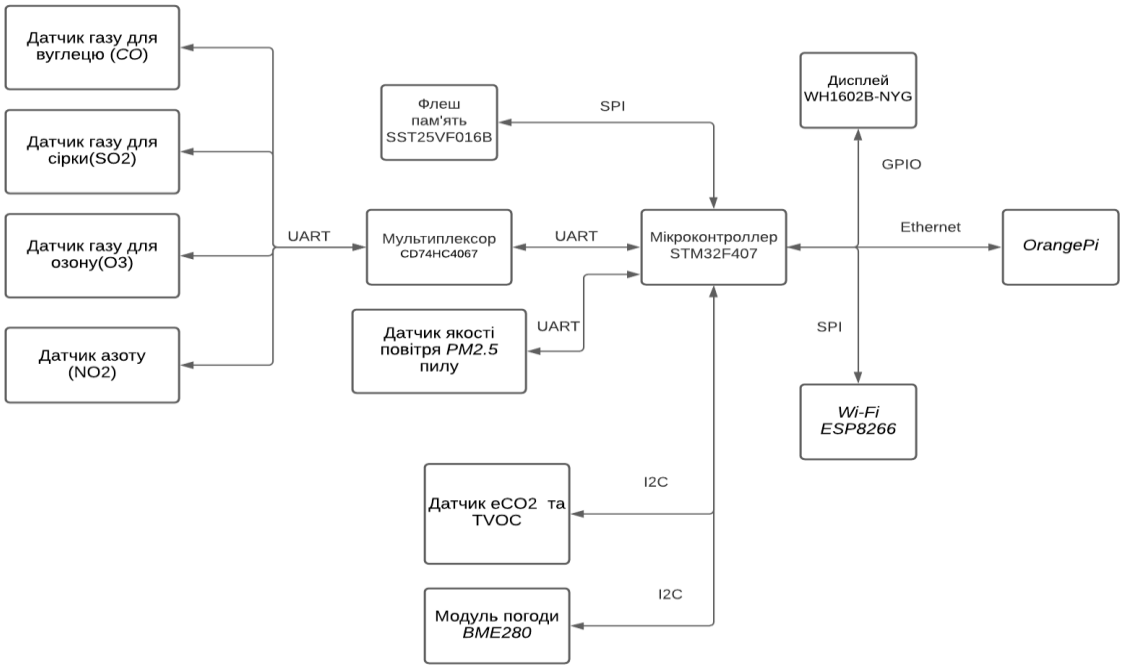

Рис.2.11. Схема підключення інтерфейсів до мікроконтролера

Для передачі даних з мікроконтролера на *OrangePi* використовується інтерфейс *Ethernet* та протокол *TCP*. На міні-комп'ютері *Orange Pi* запущенно три сервіси: *OBDGPSLogger*, *IC-Lib*, *CloudHub*.

OBDGPSLogger виконує функцію прийому даних та упаковку їх в пакет *json* і передає дані далі через лінукс сокет в *IC-Lib*. *IC-Lib* – це кластер приладів або інтерфейс людина-машина (*HMI*) пристрій, він відправляє дані через лінукс сокет на *CloudHub*. *CloudHub* збирає інформацію з усіх пристроїв, працює як сітчастий

вузол для збору повідомлень від пристроїв та інфраструктури, а потім відправляє агреговані дані на *cloud* або від хмари до певного пристрою чи вузла.

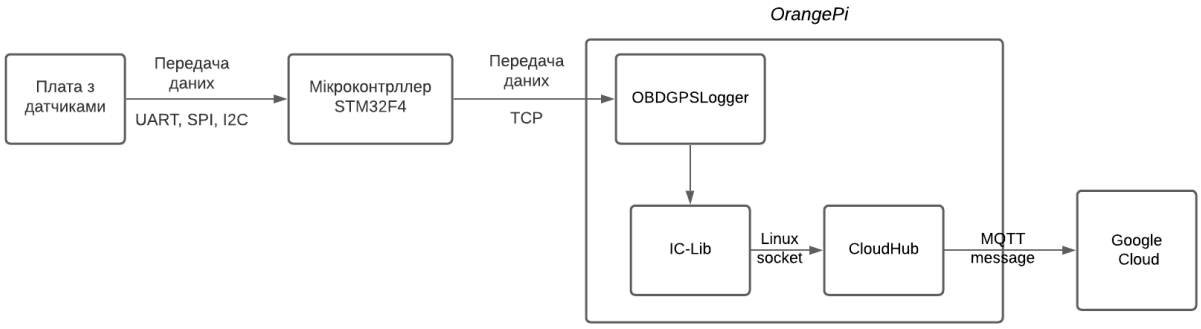

Рис.2.12. Схема передачі даних апаратно-програмного комплексу

Сітчаста мережа – це одна із топологій комп'ютерних мереж(Рис.2.13), в яких кожен вузол передає дані через мережу і виступає в ролі комутатора. Всі вузли працюють у розподілі даних в мережі, отже кожен вузол бере участь у передачі даних. Топологія забезпечує:

- Відмовостійкість мережі, що потрібно для підключення нових пристроїв;
- Надійість, адже мережа стійка до втрати окремих елементів.
- Ємність комунікаційного каналу гарантована, оскільки має більше взаємозв'язків.

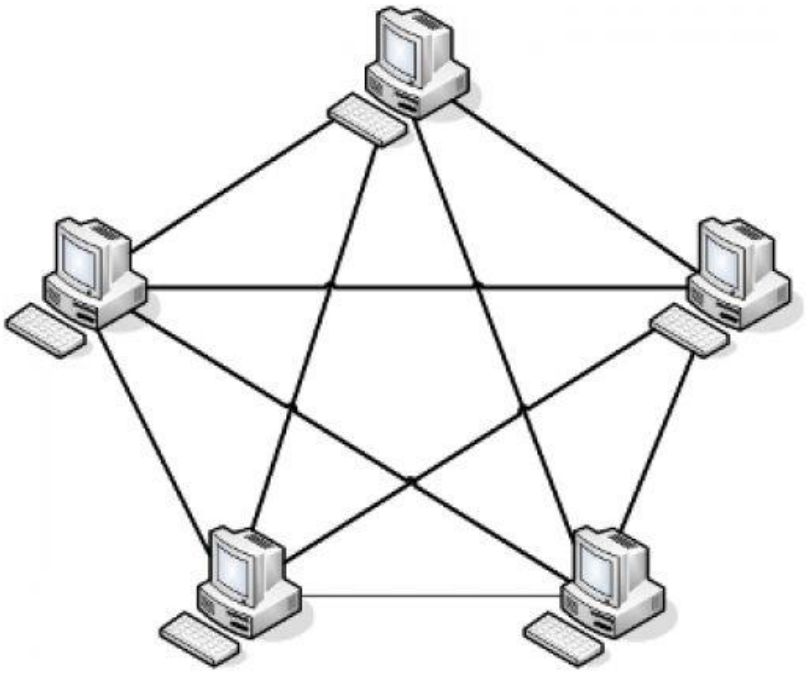

Рис.2.13. Топологія сітчастої мережі

Висновки до розділу

Використання датчиків *DGS-CO, DGS-NO2, DGS-SO2, DGS-O3, SEN0231* та *SDS011* є доцільним, адже точність вимірювання достаній для визначення рівню забрудненості, також вони використовують прості інтерфейси для роботи з ними. Також вони мають не великий розмір що дає змогу помістити їх в комлекс. Вони споживають не велику кількість енергії, що дає змогу зробити комплекс портативним. В ході проєктування комплексу було виявленно що кількість інтерфейсів *UART* у мікроконтроллера *STM32F407* не достатня для підключення всіх датчиків, проблема була вирішена за допомогою мультиплексора *CD74HC4067* який дає можливість підключити всі датчики та отримувати з них дані.

Також доцільним було використовувати мікроконтроллер *STM32F407* так як він має великий спектр інтерфейсів таких як *UART, SPI, I2C, Ethernet, GPIO*, а продуктивність є достаньою для оброблення даних в реальному часі.

Використання *Orange Pi* не є обов'язковим, замість нього може бути будь-який комп'ютер з операційною системою *Ubuntu,* на якому буде запущене все програмне забезпечення для сбору, обробки та відправки даних на *Google Cloud.*

Модуль *Wi-Fi ESP8266,* виконує функцію налаштування комлексу, та може використовуватись для відправки даних через *Wi-Fi.*

РОЗДІЛ 3
ПРОГРАМНЕ ЗАБЕЗЧЕННЯ КОМПЛЕКСУ

3.1. Програмне забезпечення мікроконтроллера *STM32F407*

При розробці програмного забезпечення для мікроконтроллера використовувалось операційна система *FreeRTOS*. *FreeRTOS* — це популярна операційна система реального часу для вбудованих систем, що була реалізована на 35 мікроконтролерах.

FreeRTOS розроблялась як проста і легка система. Основною мовою реалізації є *C*. Кількість коду, з використанням асемблера приблизно 1%.*FreeRTOS* забезпечує методи для роботи з декількома нитками або задачами, м'ютексами, семафорами і таймерами. А режим з таймером без переривань є доступний для малопотужних застосувань. Є підтримка пріоритетів ниток. На додачу існує чотири схеми виділення пам'яті:

- просте виділення пам'яті;
- виділення і звільнення із застосування дуже простого, швидкого алгоритму;
- більш складний і швидкий алгоритм виділення і звільнення пам'яті з об’єднанням пам'яті;
- застосування методів бібліотеки *C* для виділення і звільнення пам'яті із деяким захистом взаємного виконання.

Розглянемо *API* яке використовуєтсья для створення задачі яка керує сервером *Ethernet* та відправляє дані на клієнт *OBDGPSLogger* зображенно на Рис. 3.1. *xTaskCreate* створює нове завдання та додає його до списку завдань, які готові до запуску. *conFigSUPPORT_DYNAMIC_ALLOCATION* потрібно встановити значення 1 у *FreeRTOSConFig.h* або залишити невизначеним (у такому випадку він буде встановлений за замовчуванням. Якщо завдання створюється за допомогою *xTaskCreate ()*, то необхідна оперативна пам'ять автоматично виділяється з купи *FreeRTOS*. Якщо завдання створюється за допомогою *xTaskCreateStatic* (), тоді

оперативна пам'ять надається програмою запису програм, тому її можна статично виділити під час компіляції.

- *ethernetiF_input* - Вказівник на функцію введення завдання.Завдання зазвичай реалізуються у вигляді нескінченного циклу; функція, яка реалізує завдання, ніколи не повинна намагатися повернутися або вийти. Однак завдання можна видалити самі.

```
status = xTaskCreate(
        ethernetif_input,
        "ethif_in",
        ETHIF_IN_TASK_STACK_SIZE,
        (void *) netif,
        ETHIF_IN_TASK_PRIO,
        &ethif_in_handle);

configASSERT(status);
```

Рис. 3.1. Частина коду для створення задачі керування *Ethernet*

- *"ethiF_in"*, - Описова назва завдання. Це в основному використовується для полегшення налагодження, але також може використовуватися для отримання дескриптора завдань.Максимальна довжина імені завдання визначається *conFig MAX_TASK_NAME_LEN* у *FreeRTOSConFig.h.*
- *ETHIF_IN_TASK_STACK_SIZE,* - Кількість слів (а не байтів!), Які слід виділити для використання як стек завдання. Наприклад, якщо стек має 16-бітну ширину, а *usStackDepth* - 100, тоді 200 байт буде виділено для використання в якості стека завдання. Ще один приклад: якщо стек шириною 32 біти, а *usStackDepth* 400, то для використання в якості стека завдання буде виділено 1600 байт. Глибина стека, помножена на ширину стека, не повинна перевищувати максимального значення, яке може міститися у змінній типу *size_t.*
- *(void *) netiF,* - Значення, яке передається як параметр до створеного завдання. Якщо *(void *) netiF* встановлено на адресу змінної, тоді змінна все ще повинна існувати, коли створене завдання виконується - тому невірно передавати адресу змінної стека.

- *ETHIF_IN_TASK_PRIO,* - Пріоритет, при якому буде виконуватися створене завдання. Системи, що включають підтримку *MPU*, можуть додатково створити завдання в привілейованому (системному) режимі, встановивши бітовий порт *ETHIF_IN_TASK_PRIO* у *uxPrriority*. Наприклад, для створення привілейованого завдання з пріоритетом 2 встановіть для *uxPriority* значення (2 | *portPRIVILEGE_BIT*). Пріоритети стверджуються меншими за *conFig* MAX_PRIORITIES. Якщо *conFigASSERT* не визначений, пріоритети обмежуються (*conFigMAX_PRIORITIES* - 1).
- *&ethiF_in_handle* - Використовується для передачі дескриптора до створеного завдання з функції *xTaskCreate ()*. *&ethiF_in_handle* необов'язковий і може мати значення *NULL*.
- Повертає функція результат в змінну *status*. Якщо завдання було створено успішно, тоді повертається *pdPASS*. В іншому випадку повертається *errCOULD_NOT_ALLOCATE_REQUIRED_MEMORY*.

Використовувати функцію відправки даних на клієнт можуть інші задачі із інших процесів, тому використвоуються семафори, які за розділяють доступ до критичним русерсам паралельно робочих процесів. Розглянемо частину коду яка відправляє дані і використовує симафори (Рис. 3.2.) *xSemaphoreTake* - макрос для отримання семафору. Семафор повинен бути створений раніше із викликом *xSemaphoreCreateBinary* (), *xSemaphoreCreateMutex* () або *xSemaphoreCreateCounting* (). Цей макрос не можна викликати з *ISR*. *xQueueReceiveFromISR* () може використовуватися для отримання семафору зсередини переривання, якщо це потрібно, хоча це не буде звичайною операцією. Семафори використовують черги як основний механізм, тому функції певною мірою взаємодіють.

- *clients_mut* - Дескриптор семафору, який береться - отриманий під час створення семафору.
- *portMAX_DELAY* - час очікування, поки семафор стане доступним. *portMAX_DELAY* може бути використаний для перетворення цього в реальний час. Для опитування семафору може бути використаний нульовий

час. Якщо *INCLUDE_vTaskSuspend* встановлено на '1', тоді вказівка часу блоку як *portMAX_DELAY* призведе до того, що завдання буде блокуватися на невизначений час (без тайм-ауту).

```
uint32_t sender_ethernet (void *data, uint32_t size){
    uint32_t err=0;
    uint32_t result=ETH_STATUS_SEND_OK;
    for (int i = 0; i < MAX_CLIENTS; i++)
    {
        if (clients[i] != 0)
        {
            xSemaphoreTake(clients_mut, portMAX_DELAY);
            err=lwip_write(clients[i], data, size);
            xSemaphoreGive(clients_mut);
            if(err!=size)
            {
                result|=1<<i;
            }
        }
    }
    return result;
}
```

Рис. 3.2. Частина коду коду яка відправляє дані і використовує симафори

Для визначення температури використовувався датчик температури *lm335z*.Для считування даних з датчика температури *LM335Z* використвується аналого-цифровий перетворювач - пристрій, що перетворює вхідний аналоговий сигнал у дискретний код (цифровий сигнал). Для цього потрібно сконфігурувати його на *STM32*, конфігурація знаходиться на рисунку 3.3.

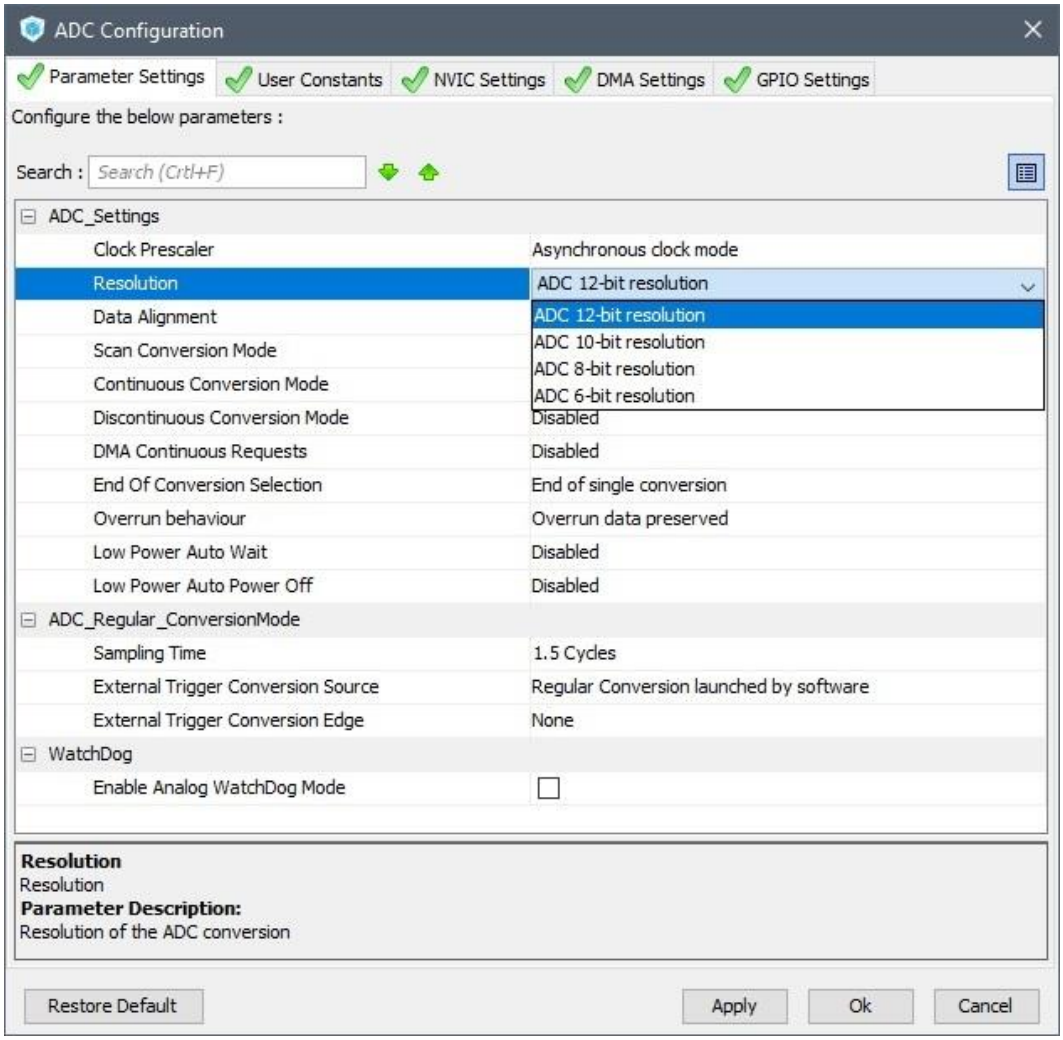

Рис. 3.3. Конфігурація *ADC* на *STM32*

Дані темпертури з датчика зберігаються за допомогою алгоритму плавучого вікна(*Floating-window algorithm*). Алгоритм плаваюче вікно - алгоритм трансформації, що дозволяє сформувати з членів тимчасового ряду набір даних, який може служити навчальним безліччю для побудови моделі прогнозування.

Під вікном у даному випадку розуміється часовий інтервал, що містить набір значень, які використовуються для формування навчального прикладу. У процесі роботи алгоритму вікно зміщується по часовій послідовності на одиницю спостереження, і кожне положення вікна утворює один приклад.

Приклад на рисунку 3.4 , якщо щотижня надходять дані про продажі протягом 50 тижнів і встановлено вікно в 5 тижнів, то в першому прикладі використовуються дані з 1 по 5 тиждень, а цільовим значенням будуть дані за 6-й тиждень. У другому випадку використовуються дані з 2 по 6 тиждень, а в якості цільового значення беруться дані за 7-ю і т.д.

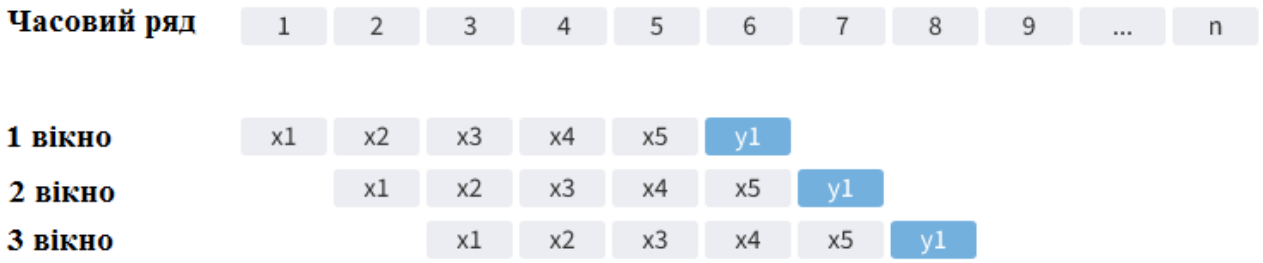

Рис 3.4. Приклад плаваючого вікна

3.2. Інструмент генерації виконуваних файлів *GNU Make*

GNU Make - це інструмент, який керує генерацією виконуваних файлів та інших не вихідних файлів програми із вихідних файлів програми. *Make* отримує свої знання про те, як побудувати свою програму з файлу, який називається *makeFile*, у якому перелічено кожен файл, що не є джерелом, і як обчислити його з інших файлів. Коли ви пишете програму, вам слід написати для неї файл *make*-файлів, щоб можна було використовувати *Make* для побудови та встановлення програми.

Можливості *Make*

Make дозволяє кінцевому користувачеві створювати та встановлювати ваш пакет, не знаючи подробиць того, як це робиться - оскільки ці деталі записуються у файл *make*, який ви надаєте.

Автоматично визначайте, які файли потрібно оновити, виходячи з того, які вихідні файли змінилися. Він також автоматично визначає належний порядок оновлення файлів, якщо один файл, що не є джерелом, залежить від іншого файлу, що не є джерелом.

Як результат, якщо ви зміните кілька вихідних файлів, а потім запустите *Make*, йому не потрібно буде перекомпілювати всю програму. Він оновлює лише ті непочаткові файли, які прямо чи опосередковано залежать від вихідних файлів, які ви змінили.

Make не обмежується жодною конкретною мовою. Для кожного файлу програми, що не є джерелом, *makeFile* вказує команди оболонки для його

обчислення. Ці команди оболонки можуть запускати компілятор для створення об'єктного файлу, компоновщик для створення виконуваного файлу.

Make не обмежується створенням пакету. Ви також можете використовувати *Make*, щоб керувати встановленням або видаленням пакета, створювати для нього таблиці тегів або будь-що інше, що ви хочете робити досить часто, щоб зробити це вартим, записуючи, як це зробити.

Правило у файлі *make* вказує "Зробити", як виконувати серію команд для побудови цільового файлу з вихідних файлів. Він також визначає список залежностей цільового файлу. Цей список повинен включати всі файли (будь то вихідні файли чи інші цілі), які використовуються як вхідні дані до команд у правилі.

Ось як виглядає просте правило:

ціль: залежності ...

команди

...

Запустивши *Make*, ви можете вказати конкретні цілі для оновлення; в іншому випадку зробіть оновлення першим цільовим переліком у файлі *make*. Звичайно, будь-які інші цільові файли, необхідні як вхідні дані для генерації цих цілей, повинні спочатку оновитись.

Make використовує *make*-файл, щоб з’ясувати, які цільові файли слід оновити, а потім визначає, які з них насправді потрібно оновити. Якщо цільовий файл новіший за всі його залежності, він уже оновлений, і його не потрібно регенерувати. Інші цільові файли потребують оновлення, але в правильному порядку: кожен цільовий файл повинен бути регенерований перед тим, як його використовувати для регенерації інших цілей.

3.3. Інструмент генерації виконуваних файлів *CMake*

CMake - це міжплатформене сімейство інструментів з відкритим кодом, розроблене для створення, тестування та упаковки програмного забезпечення. *CMake* використовується для управління процесом компіляції програмного забезпечення, використовуючи прості незалежні від платформи файли конфігурації

від платформи та компілятора, та генерує власні файли *make*-файлів та робочі області, які можна використовувати у вибраному вами середовищі компілятора. Набір інструментів *CMake* був створений *Kitware* у відповідь на потребу в потужному середовищі збірки між платформами для проектів з відкритим кодом, таких як *ITK* та *VTK*.

Найпростіший проект - це виконуваний файл, побудований із файлів вихідного коду. Для простих проектів потрібен три рядковий файл *CMakeLists.txt*. Це буде відправною точкою для нашого підручника. Створіть файл *CMakeLists.txt* у каталозі *Step1*, який виглядає так:

```
cmake_minimum_required(VERSION 3.10)
# set the project name
project(Tutorial)
# add the executable
add_executable(Tutorial tutorial.cxx)
```

Розглянемо файл *CMakeLists.txt* для прошивки мікроконтроллера(Рис.3.5). Команда *cmake_minimum_required (VERSION 3.0)* неявно викликає команду *cmake_policy (VERSION VERSION 3.0),* щоб вказати, що поточний код проекту записується для заданого діапазону версій *CMake*. Усі політики, відомі для запущеної версії *CMake* і введені у версії <*min*> (або <*max*>, якщо вказано) або раніше, будуть налаштовані на використання нової поведінки. Усі політики, введені в пізніших версіях, буде скасовано. Це фактично вимагає поведінки, переважної для даної версії *CMake*, і повідомляє новіші версії *CMake* попереджати про свої нові політики.

Команда *project(AirC_Device)* встановлює ім'я проекту та зберігає його у змінній *PROJECT_NAME*. При виклику з верхнього рівня *CMakeLists.txt* також зберігає ім'я проекту у змінній *CMAKE_PROJECT_NAME*.

Також встановлює змінні:

PROJECT_SOURCE_DIR, <PROJECT-NAME> _SOURCE_DIR

PROJECT_BINARY_DIR, <PROJECT-NAME> _BINARY_DIR

Подальші змінні встановлюються необов'язковими аргументами, описаними нижче. Якщо будь-який з цих аргументів не використовується, тоді відповідні змінні встановлюються в порожній рядок.

Команда *add_subdirectory(source_dir [binary_dir] [EXCLUDE_FROM_ALL]) додає підкаталог до збірки. Source_dir визначає каталог, в якому знаходяться вихідні файли CMakeLists.txt та коду*. Якщо це відносний шлях, він буде оцінений щодо поточного каталогу (типове використання), але це може бути і абсолютний шлях. *binary_dir* визначає каталог, в який слід розмістити вихідні файли. Якщо це відносний шлях, він буде оцінений щодо поточного вихідного каталогу, але це може бути і абсолютний шлях. Якщо *binary_dir* не вказано, буде використано значення *source_dir* перед розширенням будь-якого відносного шляху (типове використання). Файл *CMakeLists.txt* у вказаному вихідному каталозі буде негайно оброблений *CMake* перед тим, як обробка у поточному вхідному файлі продовжиться за цією командою.

Якщо наведено аргумент *EXCLUDE_FROM_ALL*, цілі у підкаталозі за замовчуванням не будуть включені в ціль ВСІ батьківського каталогу і будуть виключені з файлів проекту *IDE*. Користувачі повинні явно будувати цілі в підкаталозі. Це призначено для використання, коли підкаталог містить окрему частину проекту, яка є корисною, але не потрібною, наприклад, набір прикладів. Зазвичай підкаталог повинен містити власний виклик команди *project* (), щоб у підкаталозі було сформовано повну систему побудови Зверніть увагу, що міжцільові залежності замінюють це виключення. Якщо ціль, побудована батьківським проектом, залежить від цілі в підкаталозі, ціль залежного буде включена в систему побудови батьківського проекту для задоволення залежності.

```
1  cmake_minimum_required(VERSION 3.0)
2  project(AirC_Device)
3  add_subdirectory(lib)
4  add_subdirectory(drivers/hal/src)
5  add_subdirectory(drivers/ksz8081rnd/src)
6  add_subdirectory(drivers/wh1602/src)
7  add_subdirectory(startup)
8  add_subdirectory(src)
```

Рис 3.5. Скріншот змісту файлу *CMakeLists.txt* для прошивки мікроконтроллера

3.4. Бібліотека для створення додатків на *STM32*

HAL (*Hardware Abstraction Layer*) - це бібліотека для створення додатків на *stm32*, розроблена компанією *ST* в 2014 році. *HAL* прийшов на зміну *SPL*. Рівень драйверів *HAL* забезпечує простий, загальний набір *API* з декількома примірниками (інтерфейси прикладного програмування) для взаємодії з верхнім рівнем (програма, бібліотеки та стеки). *API* драйверів *HAL* поділяються на дві категорії: загальні *API*, які забезпечують загальні та загальні функції для всіх серій *STM32* та *API* розширень, які включають специфічні та налаштовані функції для даного номера рядка або частини.

Драйвери *HAL* містять повний набір готових до використання *API*, які спрощують реалізацію користувацьких додатків. Наприклад, периферійні пристрої зв'язку містять *API* для ініціалізації та налаштування периферії, управління передачею даних у режимі опитування, обробки переривань або *DMA* та управління помилками зв'язку. Драйвери *HAL* орієнтовані на функції, а не на *API*.

Наприклад, *API*-інтерфейси таймера поділяються на кілька категорій, що відповідають функціям *API*, такі як базовий таймер, захоплення та широтно-імпульсна модуляція (ШІМ). Рівень драйвера *HAL* реалізує виявлення помилок під час виконання, перевіряючи вхідні значення всіх функцій. Така динамічна перевірка підвищує надійність прошивки. Виявлення часу виконання також підходить для розробки користувацьких додатків та налагодження.

Драйвери *LL* пропонують апаратні послуги на основі доступних функцій периферійних пристроїв *STM32*. Ці служби точно відображають апаратні можливості та забезпечують атомні операції, які необхідно викликати, дотримуючись моделі програмування, описаної в довідковому посібнику з лінійки продуктів. Як результат, служби *LL* не засновані на автономних процесах і не потребують додаткових ресурсів пам'яті для збереження своїх станів, лічильників або покажчиків даних. Всі операції виконуються шляхом зміни змісту відповідних периферійних регістрів. На відміну від *HAL*, *API LL* не надаються для периферійних пристроїв, для яких оптимізований доступ не є ключовою функцією, або для тих, хто вимагає важкої конфігурації програмного забезпечення та / або

складного стеку верхнього рівня (наприклад, *USB*). *HAL* та *LL* доповнюють один одного і охоплюють широкий спектр вимог до застосування:

• *HAL* пропонує *API* високого рівня та орієнтовані на функції з високим рівнем переносимості. Вони приховують *MCU* та периферійну складність від кінцевого користувача.

• *LL* пропонує *API* низького рівня на рівні реєстру, з кращою оптимізацією, але меншою портативністю. Вони вимагають глибоких знань мікроконтролера та периферійних характеристик. Вихідний код драйвера *HAL* та *LL* розроблений в *Strict ANSI-C*, що робить його незалежним від інструментів розробки. Це перевіряється за допомогою інструменту статичного аналізу *CodeSonar®*. Це повністю задокументовано. Він відповідає стандарту *MISRA C®: 2004*.

3.5. Конфігурація мікроконтроллера

Конфігурація мікроконтроллера знаходиться в флеш пам'яті *SST25VF016B*. Конфігурація записується в флеш пам'ять після налаштування за допомогою веб-інтерфейсу через *Wi-Fi*, та считується кожен раз після включення мікроконтроллера. Також на флеш пам'яті знаходяться буферні дані, структура даних знаходиться на рисунку 3.6.

```
typedef struct boxConfig
{
    double latitude;
    double longitude;
    double altitude;
    unsigned long long int SO2_specSN;
    unsigned long long int NO2_specSN;
    unsigned long long int CO_specSN;
    unsigned long long int O3_specSN;
    uint8_t id;
    uint8_t working_status;
    char description[500];
    char wifi_pass[64];
    char wifi_ssid[32];
    char fb_email[32];
    char type[19];
    char ip[15];
    unsigned long long int sent_packet_counter;
}boxConfig_S;
```

Рис 3.6. Структура даних знаходиться, що знаходиться в в флеш пам'яті *SST25VF016*

- *latitude –GPS* широта*;*

- *longitude* –*GPS* довгота;
- *altitude* –*GPS* висота;
- *SO2_specSN* – дані про кількість *CO2* в повітрі;
- *NO2_specSN* – дані про кількість *NO2* в повітрі;
- *CO_specSN* – дані про кількість *CO* в повітрі;
- *O3_specSN* – дані про кількість O3 в повітрі;

- *id* – унікальний індифікатор комплексу;
- *working_status* – статус роботи комплексу;
- *description* – опис комплексу;
- *wiFi_pass* – пароль від *Wi-Fi*;
- *wiFi_ssid* – унікальне найменування бездротової мережі;
- *Fb_email* – почтовий електронний адрес;
- *type* – тип комплексу;
- *ip* – IP-адреса комплексу.
- sent_packet_counter – номер відправленого пакету.

Ця структура обернена в структуру з *CRC16 CCITT*(Рис. 3.7.) – це алгоритм знаходження контрольної суми, призначений для перевірки цілісності даних. *CRC* є практичним застосуванням завадостійкого кодування, заснованим на певних математичних властивості циклічного коду.

```
typedef struct boxConfig_CRC16_CCITT
{
    boxConfig_S cfg;
    unsigned short CRC16_CCITT;
}boxConfig_CRC16_CCITT_S;
```

Рис. 3.7. Структура даних обернена в структуру з *CRC16 CCITT*

Алгоритм *CRC* базується на властивості ділення з остачею двійкових многочленів, тобто многочленів над скінченним полем *GF*(2). Значення *CRC* є по суті остачею від ділення многочлена, відповідного вхідним даним, на деякий фіксований породжувальний многочлен.Кожній послідовності бітів a_0, a_1, … , a_{N-1} взаємно однозначно зіставляється двійковий $\sum_{n=0}^{N-1} a_n x^n$, послідовність коефіцієнтів якого являє собою початкову послідовність.

3.6. Структура данних пакету, який відправляє *STM32F407*

Всі дані з датчиків збираються за допомогою задачі *data_collector*. Задача кожні 30 секунд опитує по черзі все датчики та отримує дані з них та заповнює камеру для повітря за допомогою вентиляторів. На рисунку 3.8 зображена структура.

- *device_id* - унікальний індифікатор комплексу;
- *device_working_status* – статус роботи комлексу;
- *device_message_counter* – лічильник повідомлень;
- *device_type* – тип дивайсу (стаціонарний або встановленний в автомобілі);
- *message_date_time* – час відправки повідомлення;
- *latitude –GPS* широта*;*
- *longitude –GPS* довгота*;*
- *altitude –GPS* висота;
- *temp_internal* – температура з датчику *LM335Z*;
- *temp* – температура з модулю *BME280;*
- *humidity* – вологість з модулю *BME280;*
- *co2* – дані про кількість *CO2* в повітрі
- *tvoc* – дані про кількість *TVOC* в повітрі
- *pressure* – дані про тиск
- *co* – дані про кількість *CO* в повітрі
- *no2* – дані про кількість *NO2* в повітрі
- *so2* – дані про кількість *SO2* в повітрі
- *o3* – дані про кількість *O3*в повітрі
- *hcho* – дані про кількість *HCHO* в повітрі
- *pm2_5* – дані про кількість *PM2,5* в повітрі
- *pm10* – дані про кількість *PM10* в повітрі

```
typedef struct dataPacket
{
    uint8_t device_id;
    uint8_t device_working_status;
    uint32_t device_message_counter;
    char device_type[19];
    char device_description[500];
    char message_date_time[24];
    double latitude;
    double longitude;
    double altitude;
    double temp_internal;
    double temp;
    double humidity;
    double co2;
    double tvoc;
    double pressure;
    double co;
    double co_temp;
    double co_hum;
    double co_err;
    double no2;
    double no2_temp;
    double no2_hum;
    double no2_err;
    double so2;
    double so2_temp;
    double so2_hum;
    double so2_err;
    double o3;
    double o3_temp;
    double o3_hum;
    double o3_err;
    double hcho;
    double pm2_5;
    double pm10;
} dataPacket_S;
```

Рис 3.8. Структура пакету

3.7. Обробка та відправка даних на *Google Cloud*

Дані з мікроконтроллера приходят клієнту *OBDGPSLogger*, клієнт відправляє дані на кластер *IC-Lib*. Кластер в свою чергу збирає дані з всіх підключених до нього клієнтів, виводить цю інформацію(Рис 3.9) та відправляє на *CloudHub*.

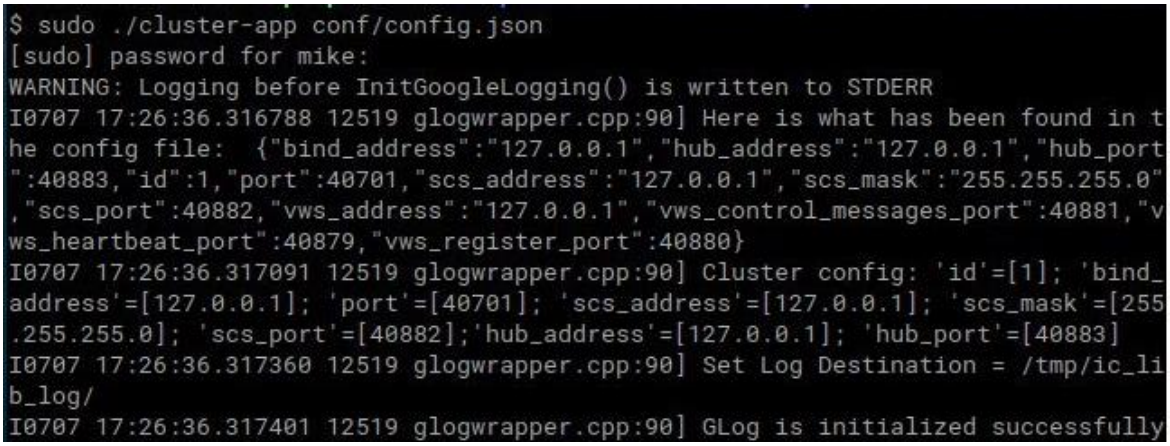

Рис 3.9. Інформація з модуля *IC-Lib*

CloudHub в свою чергу створює безпечене підключення до *Google Cloud* за допомогою шифрування. *CloudHub* має такі конфігураційні параметри таблиця 3.1

Таблиця 3.1

CloudHub конфігураційні параметри

Назва параметру	Опис
1	2
mqtt_msg_parallel_instances	Кількість максимальних дочірніх процесів
mqtt_msg_operation_timeout	Час очікування операції зв'язку *mqtt* (якщо дочірній процес не зберігає дані у цьому часовому вікні *MqttMsgManager* вбиває дочірній процес, щоб звільнити слот для наступного об'єкта)
data_processos_group_intervel	Інтервал для даних групи містить 2 поля: сек - секунди та мілісекунди мс
data_processor_store_every_pkt	Хмарний концентратор може видалити деякі пакети з групи, якщо пакети надходять надто часто. Якщо Ви хочете зберегти кожен пакет, встановіть для цього параметра значення 1. Якщо ви хочете скинути кілька пакетів, встановіть, який пакет буде доданий до групи.Приклад: *data_processor_store_every_pkt* = 3, лише кожні 3 пакети будуть зберігатися в хмарі

Продовження табл. 3.1

1	2
port	Порт прослуховування *udp*
mqtt_group_param	Цей розділ містить параметри, що залежать від *mqtt*
mqtt_group_param->registry_id	Ідентифікатор хмарного реєстру
mqtt_group_param->project_id	Ідентифікатор хмарного проєкту
mqtt_group_param->root_cert	Хмарний кореневий сертифікат
mqtt_group_param->objects->mqtt_id	Ідентифікатор об'єкта *MQTT*, зареєстрований у хмарі
mqtt_group_param->objects-> sbc_id	Ідентифікатор об'єкта *SBC* (цей ідентифікатор повинен відповідати ідентифікатору сервера емуляції та / або ідентифікатору *SmartCityIC*)
mqtt_group_param->objects-> key_path	Кожен об’єкт *mqtt* має власний приватний ключ (відкритий ключ зареєстрований у хмарі)
mqtt_group_param->objects-> key_algorytm	Ключ об’єкта *mqtt* який генерує алгоритм

Для шифрування даних можливо використовувати алгортим *RSA* або *AES*. *RSA* — це криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі факторизації великих цілих чисел.

RSA став першим алгоритмом такого типу, придатним і для шифрування, і для цифрового підпису. *AES* — симетричний алгоритм блочного шифрування (розмір блока 128 біт, ключ 128/192/256 біт. *AES* має фіксовану довжину у 128 біт, а розмір ключа може приймати значення 128, 192 або 256 біт. Через фіксований розмір блоку *AES* оперує із масивом 4×4 байт, що називається станом (версії алгоритму із більшим розміром блоку мають додаткові колонки).

Дані з *CloudHub* потрапляють до *Google Cloud*. *Google Cloud* — це потужна хмарна платформа, яка покращує управління даними та зменшує додаткові витрати на управління інфраструктурою, підтримку серверів і налаштування мереж. *Google Cloud* надає доступ до серверів, мереж і дисків 24 години на добу з будь-якої точки земної кулі, не хвилюючись за питання безпеки, амортизації або електроенергії.

В *Google Cloud* використовується *Firebase Realtime Database*. *F*irebase надає в режимі реального часу базу даних та бекенд як службу. База даних *Firebase Realtime* - це база даних, розміщена в хмарі. Дані храняться у форматі *JSON* та синхронізуються в реальному часі з кожним підключеним клієнтом. База даних являється *NoSQL* .

NoSQL — база даних, яка забезпечує механізм зберігання та видобування даних відмінний від підходу таблиць-відношень в реляційних базах даних. Відрізніється вона від *SQL* тим що надає: простоту дизайну схеми БД, значно спрощене горизонтальне масштабування на кластери машин (що є проблемою для реляційних баз даних), і тонкий контроль над доступністю. Структури даних, що використовуються в *NoSQL* (такі як ключ-значення, сховище з широким стовпчиком, граф, документ) є відмінними від тих, що використовуються за замовчуванням в реляційних базах, що робить тим самим деякі операції над даними значно швидшими на *NoSQL*. Точна відповідність використання *NoSQL* бази даних залежить від проблем, які треба вирішити. Іноді структури даних, які використовуються в *NoSQL* базах можуть розглядатись як більш гнучкі ніж таблиці реляційних моделей.

Дані які відправляє CloudHub до *Firebase Realtime Database,* оброблються за допомогою мови програмування *Python*. *Python* – це інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. За допомогою великою кількості бібліотек швидко та эфективно оброблюються дані та зберігаються в базі даних (Рис 3.10.).

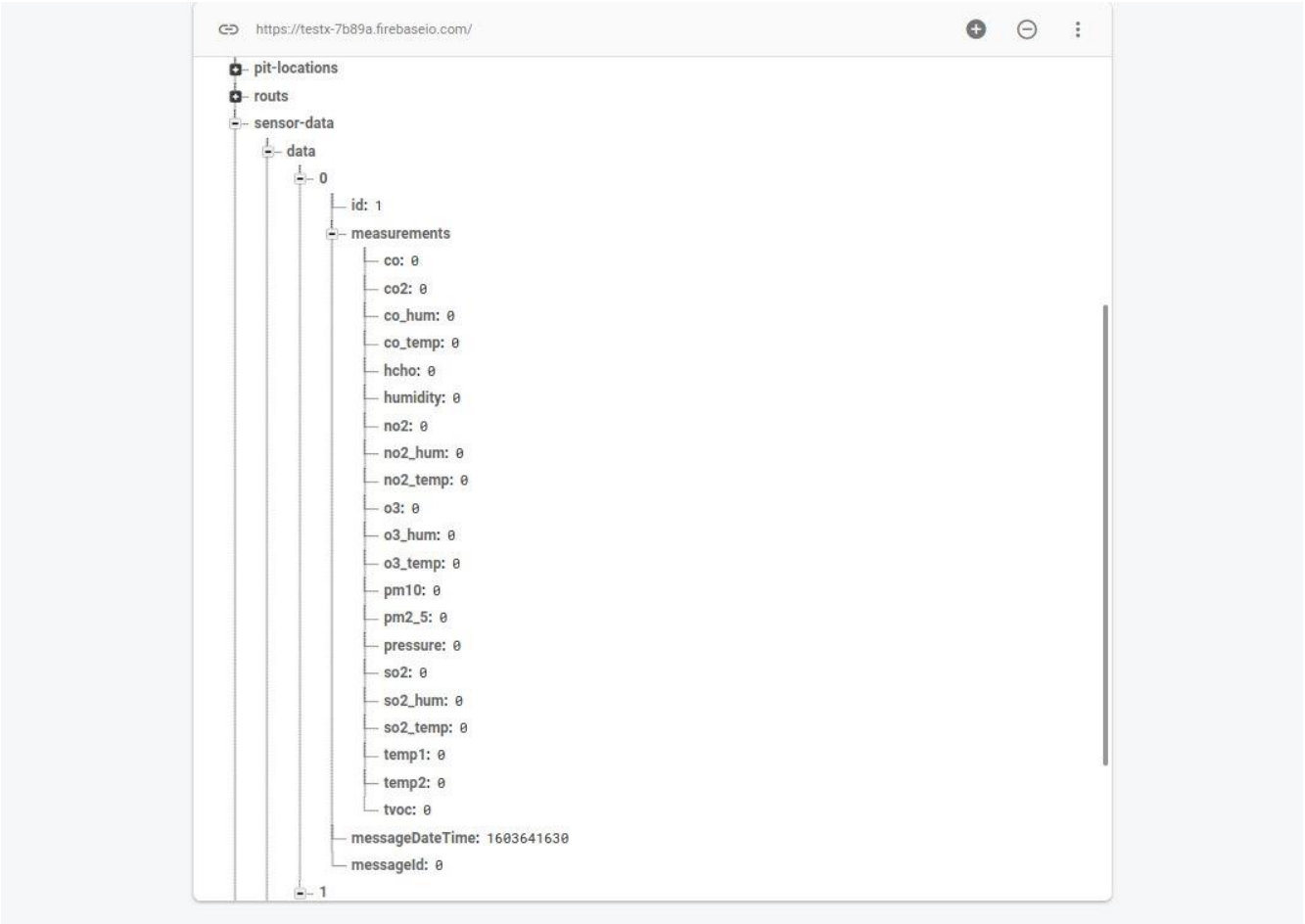

Рис. 3.10. Скріншот інформації збереженої в *Firebase Realtime Database*

3.8. Аналіз даних та оцінка якості повітря

Дані збережені в базі даних, аналізуються за допомогою *Python* функцій за допомогою функцій. Комплес оцінює якість повітря за допомогою *AQI*(*air quality index*). *AQI* - це індекс якості повітря, ця абревіатура використовується в усіх світових екологічний державних органах для інформування громадськості про рівень забруднення повітря а так само прогнозування забруднення повітря. У випадки збільшення рівня індексу якості повітря, на великий відсоток забруднення, суспільство отримує значний вплив на здоров'я. Різні країни мають свій власний індекс якості повітря відповідає національним стандартам.

Обчислення індексу якості повітря вимагає отримання інформація про рівень забруднення за певний середній період отриманий з газоаналізаторів моніторингу якості повітря або отриманих розрахунковим методом (що є менш точний метод визначення забруднення повітря). Беруться разом концентрація і час поширення

забруднюючих речовин в атмосфері. Ефект впливу на здоров'я конкретного обсягу забруднень визначається епідеміологічними дослідженнями. Забруднювачі повітря розрізняються по силі і функції використовуваної для того, що б конвертувати забруднювач повітря в індекс якості повітря варіюється по забруднювачі. Індекс якості повітря зазвичай формується за рівнями, кожен рівень має свій опис і характеристику, колірний код і стандартизоване інформаційне повідомлення про вплив на суспільне здоров'я.

Індекс якості повітря може збільшиться з багатьох факторів таких як, дорожній рух в годину пік, під час пожеж, відсутності вітру, або через брак розріджувачів забруднювачів повітря. Нерухоме повітря часто викликаний антициклоном, інверсією температури або слабкою швидкістю вітру дозволяє залишатися забруднювачів повітря в одному місці, що призводить до високої концентрації забруднюючих речовин, а так само хімічною реакцією між забруднюючими речовинами в атмосфері а так само призводить до зможу.

Існують категорії індексу якості повітря(Рис. 3.11).

Категорія AQI (Діапазон)	PM_{10} (24год)	$PM_{2.5}$ (24год)	NO_2 (24год)	O_3 (8год)	CO (8год)	SO_2 (24год)	NH_3 (24год)	Pb (24год)
Хороший (0-50)	0-50	0-30	0-40	0-50	0-1.0	0-40	0-200	0-0.5
Задовільний (51-100)	51-100	31-60	41-80	51-100	1.1-2.0	41-80	201-400	0.5-1.0
Помірно забруднений (101-200)	101-250	61-90	81-180	101-168	2.1-10	81-380	401-800	1.1-2.0
Високий (201-300)	251-350	91-120	181-280	169-208	10-17	381-800	801-1200	2.1-3.0
Дуже високий (301-400)	351-430	121-250	281-400	209-748	17-34	801-1600	1200-1800	3.1-3.5
Небезпечний (401-500)	430+	250+	400+	748+	34+	1600+	1800+	3.5+

Рис. 3.11. Категорії індексу якості повітря

Індекс якості повітря є кусково-лінійною функцією від концентрації забруднення. Для перетворення концентрації індексу використовується таке рівняння:

$$I = \frac{I_{high} - I_{low}}{C_{high} - C_{low}} (C - C_{low}) + I_{low}$$, де

I = індекс якості повітря,

C = концентрація забруднювальної речовини,

C_{low}= концентрація зупинки, ≤ ,

C_{high} = концентрація зупинки, ≥ ,

I_{low}= точка зупинки індексу(Рис 3.12.) , що відповідає,

I_{high} = точка зупинки індексу(Рис 3.12.), що відповідає.

O_3 (ppb)	O_3 (ppb)	$PM_{2.5}$ (µg/m^3)	PM_{10} (µg/m^3)	CO (ppm)	SO_2 (ppb)	NO_2 (ppb)	AQI	AQI
C_{low} - C_{high} (avg)	C_{low} - C_{high} (avg)	C_{low}- C_{high} (avg)	C_{low} - C_{high} (avg)	C_{low} - C_{high} (avg)	C_{low} - C_{high} (avg)	C_{low} - C_{high} (avg)	I_{low} - I_{high}	Категорія
0-54 (8-hr)	-	0.0-12.0 (24-hr)	0-54 (24-hr)	0.0-4.4 (8-hr)	0-35 (1-hr)	0-53 (1-hr)	0-50	Добрий
55-70 (8-hr)	-	12.1-35.4 (24-hr)	55-154 (24-hr)	4.5-9.4 (8-hr)	36-75 (1-hr)	54-100 (1-hr)	51-100	Задовільний
71-85 (8-hr)	125-164 (1-hr)	35.5-55.4 (24-hr)	155-254 (24-hr)	9.5-12.4 (8-hr)	76-185 (1-hr)	101-360 (1-hr)	101-150	Шкідливий для групи ризику
86-105 (8-hr)	165-204 (1-hr)	55.5-150.4 (24-hr)	255-354 (24-hr)	12.5-15.4 (8-hr)	186-304 (1-hr)	361-649 (1-hr)	151-200	Шкідливий
106-200 (8-hr)	205-404 (1-hr)	150.5-250.4 (24-hr)	355-424 (24-hr)	15.5-30.4 (8-hr)	305-604 (24-hr)	650-1249 (1-hr)	201-300	Дуже шкідливий
-	405-504 (1-hr)	250.5-350.4 (24-hr)	425-504 (24-hr)	30.5-40.4 (8-hr)	605-804 (24-hr)	1250-1649 (1-hr)	301-400	Небезпечний
-	505-604 (1-hr)	350.5-500.4 (24-hr)	505-604 (24-hr)	40.5-50.4 (8-hr)	805-1004 (24-hr)	1650-2049 (1-hr)	401-500	

Рис. 3.12. Точки зупинки індексу

В реальному часі дані безперервного моніторингу, як правило, доступні у вигляді середнього значення за одну годину. Однак, обчислення індексу якості повітря для деяких забруднювальних речовин, до прикладу озону, вимагає виведення середнього значення за кілька годин даних. Для точного відображення поточної якості повітря кількагодинне середнє значення, яке використовується для обчислення індексу, повинне бути зосереджене на поточному часі, але оскільки концентрація за майбутні години невідома і важко зробити точну оцінку, *EPA* використовує замінник значення концентрації для оцінки цих кількох годин.

Оброблені дані можливо переглянути на веб-сайті. На веб-сайті знаходиться карта на якій для кожної *GPS* точки створюється маркер, на якому знаходиться індекс якості повітря, якщо вибрати маркер можливо отримати детальну інформацію про кожен параметр повітря(Рис.3.13).

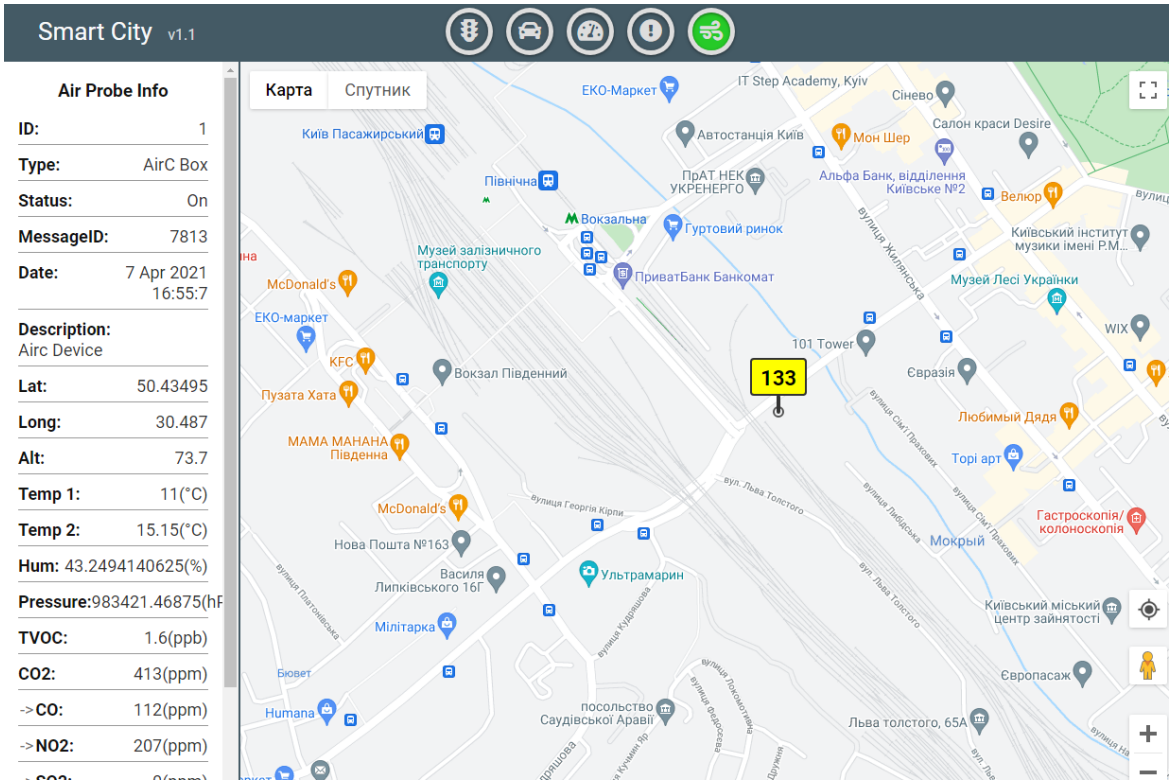

Рис.3.13. Скріншот сайту з даними

Висновки до розділу

У третій частині дипломного проєкту було обранний спосіб автоматичного збору програмної частини, що дозволило швидко вести розробку програмної частини комплексу. Також було обранно стурктуру пакету даних яка відправляється на сервер. Обрано мови програмування для швидкої та зручної розробки. Визначили спосіб шифрування даних для забезпечення надійності відправки даних на сервер *Google Cloud*.

Розроблено програмне забезпечення для оцінки якості повітря, на основі індексу якості повітря та точок зупину індексу.

ВИСНОВКИ

У дипломному проєкті на основі існуючих апаратно-програмних комплексів було розроблено апаратно-програмний комплекс контролю якості повітря.

У першому розділі розглянуто сучасні існуючі комплекси для визначення яксоті повітря. Серед них було обрані характерстики повітря які потрібно аналізувати для оцінки якості. Також було обрано метод зберігання даних, та визначенно що використати аккамулятора не доцільно, адже комплекс буде стаціонарним.

У другій частині розглягнуто датчики *DGS-CO, DGS-NO2, DGS-SO2, DGS-O3, SEN0231* та *SDS011* для аналізу повітря та мікроконтроллер *STM32F407F*. Було вирішено використати їх в проєктуванні комплекса, адже вони мають зручні інтерфейси комунікації. Також спроєктована структура даних комунікації модулів між собою.

У третьому розділі дипломного проєкту було розроблено програмне забезпечення для комунцікації, обробки, та відправки даних на веб-сервер. На основі отриманих даних веб-сервер аналізує та оцінює якість повітря в точці де знаходиться комплекс.

Апаратно-програмний комплекс спроєктований повністю та створений прототип, який було протестовано та налагоджено для отримання оцінки якості повітря. Завдяки комплексу можна попередити вразливих людей, які страждають серцево-судинними або респіраторними хворобами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Таненбаум Э.С. Компьютерные сети. 5-е изд. – М.: Питер, 2019.- 960с.
2. Таненбаум Э.С. Архитектура компьютера. 6-е изд. – М.: Питер, 2018.- 816с.
3. Шпак З.Я. Програмування мовою С. – М.: Оріяна-Нова, 2006.- 432с.
4. Muhammad Ali Mazidi, Shujen Chen, Eshragh Ghaemi.STM32 Arm Programming *F*or Embedded Systems: Using C Language with STM32 Nucleo – M.: MicroDigitalEd, 2018. – 377 с.

Електронні Internet-ресурси:

5. STM32*F*407VG Datasheet. Режим доступу : https://www.st.com/resource/en/datasheet/stm32*F*407vg.pdf (дата звернення: 01.06.2021).
6. STM32*F*407VG TRM. Режим доступу : https://www.st.com/resource/en/re*F*erence_manual/dm00031020-stm32*F*405415-stm32*F*407417-stm32*F*427437-and-stm32*F*429439-advanced-armbased-32bit-mcus-stmicroelectronics.pdf (дата звернення: 01.06.2021).
7. STM32F4DISCOVERY UM. URL: https://www.st.com/resource/en/user_manual/dm00039084-discovery-kit-with-stm32*F*407vg-mcu-stmicroelectronics.pdf (дата звернення: 01.06.2021).
8. GL StarterKit Circuit Diagram. Режим доступу :

   https://github.com/GlobalLogicEdu/GL-SK-BSP/blob/master/Documentation/Schematic_1.1.pdf (дата звернення: 01.06.2021).
9. Модуль вимірювання CO цифровий. Режим доступу :

   https://www.spec-sensors.com/wp-content/uploads/2017/01/DGS-CO-968-034.pdf (дата звернення: 01.06.2021).
10. Модуль вимірювання NO2 цифровий. Режим доступу :

https://www.spec-sensors.com/wp-content/uploads/2017/01/DGS-NO2-968-043_9-6-17.pdf (дата звернення: 01.06.2021).

11. Модуль вимірювання SO2 цифровий. Режим доступу :

https://www.spec-sensors.com/wp-content/uploads/2017/01/DGS-SO2-968-038.pdf (дата звернення: 01.06.2021).

12. Модуль вимірювання O3 (озон) цифровий. Режим доступу :

https://www.spec-sensors.com/wp-content/uploads/2017/01/DGS-O3-968-042_9-6-17.pdf (дата звернення: 01.06.2021).

13. CCS811 Air Quality Sensor. Режим доступу : https://wiki.d*F*robot.com/Gravity:%20CCS811%20Air%20Quality%20Sensor%20SKU:%20SEN0318 (дата звернення: 01.06.2021).

14. Комплексний модуль вимірювання органічних газів - CH3(толуол), C6H6(бензол), H2CO(формальдегід) . Режим доступу :

https://wiki.d*F*robot.com/Gravity__HCHO_Sensor_SKU__SEN0231#target_4 (дата звернення: 01.06.2021).

15. Модуль вимірювання температури, вологості, тиску. Режим доступу :

https://cdn-shop.ada*F*ruit.com/datasheets/BST-BME280_DS001-10.pdf (дата звернення: 01.06.2021).

16. C++11 standard. Режим доступу :

https://ru.wikipedia.org/wiki/C%2B%2B11 (дата звернення: 01.06.2021).

17. Комплексний модуль вимірювання органічних газів - CH3(толуол), C6H6(бензол), H2CO(формальдегід) . Режим доступу :

https://wiki.d*F*robot.com/Gravity__HCHO_Sensor_SKU__SEN0231#target_4 (дата звернення: 01.06.2021).

18. Індекс якості повітря. Режим доступу :

https://uk.wikipedia.org/wiki/Індекс_якості_повітря (дата звернення: 01.06.2021).

19. Uart - послідовний інтерфейс передачі даних. Режим доступу :

http://jak.bono.odessa.ua/articles/uart-poslidovnij-inter*F*ejs-peredachi-danih.php (дата звернення: 01.06.2021).