DOI 10.31891/2307-5732-2019-279-6-184-191
УДК 504.064

А.В. КОВАЛЬ, А.Г. ТКАЧУК, М.С. ГРИНЕВИЧ
Державний університет «Житомирська політехніка»
Т.Л. КОВАЛЬ
Житомирський національний агроекологічний університет

МОБІЛЬНА БЕЗДРОТОВА СИСТЕМА ДЛЯ АНАЛІЗУ ЯКОСТІ ПОВІТРЯ

У роботі розглянуто реалізацію одного зі способів моніторингу та аналізу якості повітря з використанням безпілотного літального апарата. В результаті аналізу отриманих експериментальних даних було обрано основу для даної системи та проведено ряд досліджень.

Ключові слова: БПЛА, якість повітря, система моніторингу, мобільна система.

A.V. KOVAL, A.H. TKACHUK, M.S. HRYNEVYCH
State University «Zhytomyr Polytechnic»
T.L. KOVAL
Zhytomyr National Agroecological University

MOBILE WIRELESS SYSTEM FOR AIR QUALITY ANALYSIS

The purpose of this work is to develop an automated system for air quality monitoring and analysis based on the mini unmanned aerial vehicle. An automated mobile wireless air quality monitoring system has been developed as an alternative way of finding contaminants to replace ground based systems. It provides more flexible use than terrestrial systems. The main task is to improve and create an alternative system that will perform the same tasks as terrestrial systems, but will be faster and able to work in inaccessible places. The system collects pollution data and searches for gas leaks in various potentially hazardous locations. This system is compact, lightweight, capable of being mounted on any unmanned aerial vehicle, capable of operating in the absence of GPS signal and capable of rapid deployment and piloting by operator or offline without endangering human life. This project is a system based on a conventional commercial unmanned aerial vehicle of any model, equipped with a gas concentration sensor and controlled by the Robot Operating System (ROS). In the course of this project all problems related to stability of measurement data were solved. The system is now capable of providing consistently high quality measurement. Some measurement experiments are needed to study the relationship between the distance from a gas source to the measurement system and the measured gas concentration. The influence of UAV parameters, such as battery level, on the measurement values is investigated.

Keywords: UAV, air quality, monitoring system, mobile system.

Вступ

В умовах сьогодення людство намагається зробити своє життя максимально простим і безпечним, тож питання безпеки є досить актуальним. Саме тому відбувається розробка різноманітних засобів та пристроїв які можуть замінити людину в умовах що є небезпечними для здоров'я або життя. Наприклад, існує багато роботів і безпілотних літальних апаратів (БПЛА) які знаходять безліч можливостей для застосування. На сьогоднішній день БПЛА широко використовуються людьми для створення фотографій з висоти, доставки їжі, а також у різних військових цілях [1]. Завдяки своїм властивостям та чудовій прохідності БПЛА можливо використовувати для вимірювання якості повітря на певний території, або для перевірки на наявність шкідливих або вибухонебезпечних газів. Саме цей спосіб застосування описано в даній статті.

Цей метод має ряд переваг, оскільки БПЛА здатні переміщуватись з досить великою швидкістю, а отже здатні дослідити набагато більшу територію, ніж здатна зробити це людина за той самий проміжок часу. Також цей спосіб виявлення витоку газу є значно дешевшим, оскільки зникає необхідність оплачувати людську працю.

Звісно, щоб виконувати поставлені задачі, БПЛА повинен відповідати ряду критеріїв. Дана система повинна бути легкою, компактною, працювати в середовищах, де неможливо використовувати GPS [2]. Крім того, є необхідними й такі функції, як робота в умовах слабкого GPS сигналу, або повної його відсутності (поблизу мостів та поблизу будівель). Також система повинна бути простою для використання та мати зв'язок з оператором з мінімальним ризиком для його здоров'я або працювати в автоматичному режимі.

Експериментальна частина

ОГЛЯД ОСНОВНИХ СКЛАДОВИХ ПРОЕКТУ. БПЛА AR. Drone Parrot v2. AR.Drone Parrot v2, як і його попередник, підтримує внесення деяких поліпшень, які можуть бути корисними для подальших досліджень. І, крім того, його вдосконалили його розробники. Зокрема встановлено нову фронтальну камеру з роздільною здатністю HD 720p і датчиком тиску. На додаток до цього, він отримав багато інших поліпшень, які здавалися незначними, на перший погляд, але при ближчому огляді вони є доволі корисними. Деякі поліпшення програмної складової і датчик тиску збільшують стабільність польоту і здатність протидії зовнішнім впливам у випадку сильного вітру. Краща камера позитивно вплинула на стабільність і точність автоматичного польоту. Потрібно відзначити наявність порту USB 2.0. Це дозволяє підключати додаткові пристрої USB без виконання апаратної модифікації основної плати.

ROS та інші програмні компоненти. *Robot Operating System (ROS)* є потужним програмним забезпеченням для керування роботами.

В даний час він активно використовується в численних дослідницьких проектах, оскільки це модульна система з відкритим кодом, яка підтримує різні мови програмування та надає потужні інструменти візуалізації [3].

Опис основних пакетів ROS: Ardrone_autonomy – пакет був створений для забезпечення зв'язку між дроном і комп'ютером з попередньо налаштованим ROS середовищем. Це дозволяє отримувати всі дані з датчиків БПЛА та надсилати команди керування від базової станції з ROS до БПЛА [4].

Tum_ardrone пакунок забезпечує можливість автономної навігації в попередньо невідомому середовищі за умов відсутності сигналу GPS орієнтуючись відповідно до зображення з бортових камер БПЛА. Також зберігається можливість ручного керування БПЛА, якщо це необхідно [5].

Пакунок калібрування камери – це спеціальний компонент ROS, який дозволяє відносно просто та швидко калібрувати монокулярні або стереокамери за допомогою спеціального зображення [6].

Пакет Ethzasl ptam використовується для калібрування камери для наступного використання PTAM [7].

РЕАЛІЗАЦІЯ. Модуль вимірювання рівня забруднення. Arduino Yún mini (рис. 1) – це розробницька плата, побудована на базі об'єднання мікроконтролеру ATmega32u4 та мікропроцесору Atheros AR9331. Atheros мікропроцесор зробив можливим запуск Linux дистрибутиву OpenWrt, що називається Linino OS. Дана розробницька плата оснащена мережевими інтерфейсами Ethernet та Wi-Fi, USB-A портом, micro-SD картковим слотом, 20 цифровими входами та виходами (7 з яких мають підтримку ШИМ і 12 можуть бути задіяні як аналогові входи), 16 MHz кварцовим резонатором, micro USB роз'ємом, підтримкою ICSP [8].

Основною причиною використання саме цієї плати є можливість забезпечення мережевого підключення через Wi-Fi. Для цієї плати було написано спеціальне програмне забезпечення для підтримки протоколу передачі даних TCP/IP. Підтримка клієнта TCP/IP дозволяє підключатися до віддалених хостів або апаратних засобів для читання і запису даних. У поточному проекті TCP/IP протокол використовується для передачі даних підключених датчиків до інших (наземних) компонентів системи.

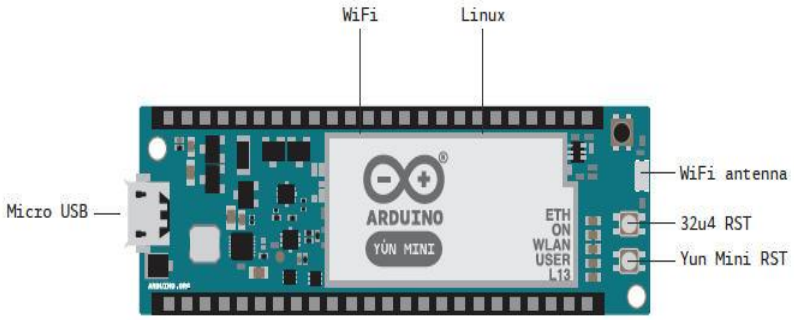

Рис. 1. Arduino Yun mini [8]

Датчик температури та вологості. Датчик SHT10 (рис. 2) об'єднує чутливий елемент і схему обробки сигналів на міні-друкованій платі і забезпечує стабільний цифровий сигнал на виході. Датчик включає компоненти, чутливі до вологості і до зміни температури, також вони знаходяться на одному чипі, з 14 бітним ЦАП перетворювачем і схемою послідовного інтерфейсу для реалізації безшовного з'єднання. Таким чином, цей датчик має відмінну якість, швидку реакцію, достатню здатність протидії завадам, високу ефективність. Він використовує двопровідний інтерфейс зв'язку (I2C) і дозволяє легко інтегруватися в хост-систему [9]. Цей датчик використовувався для тестування програми побудови карти витоку газу замість газового датчика. Оскільки тести з газом потребують особливих умов і є доволі небезпечними, тестування програмних компонентів виконувалося без газу.

Рис. 2. SHT10 датчик

Рис. 3 демонструє AR.Drone v2 з встановленими Arduino Yun mini та SHT10 датчиком.

Рис. 3. Arduino Yun та SHT 10 встановлені на шасі AR. Drone Parrot v2

ROS робоче оточення та додаткові ROS пакунки. Можливість перегляду інформації про виміри рівня газу в ROS забезпечується власно розробленим пакетом в ROS. Цей пакет ROS включає в себе два ROS сервіси. Одна з цих служб використовується як видавець, а інший, як підписник (ця інформація більш докладно описана в ROS wiki [10]). ROS містить багато вузлів, і майже всі функціональні можливості забезпечуються вузлами, до того ж вузол – це загальне ім'я якоїсь структури, яку можна використовувати для різних завдань, і ці вузли з'єднані один до одного для реалізації заздалегідь визначених функціональних можливостей. Для отримання деякої інформації ззовні в ROS потрібно створити повідомлення (детально [10]) і використовувати невелику програму, яка називається видавцем. Найважливішим функціоналом видавця є отримання деякої інформації ззовні та надання такої інформації доступним для інших вузлів ROS.

Вимірювальна плата, яка вбудовується в AR.Drone v2 - Arduino Yun, має власний модуль Wi-Fi, тому він підключається безпосередньо до ПК наземної станції. AR.Drone v2 підключений до того ж комп'ютера через Wi-Fi. Для підключення AR.Drone Parrot v2 і Arduino Yun до того ж Wi-Fi хоста спеціальний скрипт був попередньо встановлений на головну плату AR.Drone, що дозволило підключати AR.Drone до існуючої точки доступу Wi-Fi замість створення власної Wi-Fi точки доступу, як це відбувалося за замовчуванням.

Таким чином вимірювальна система стала цілком незалежною від типу платформи носія, оскільки не має жодних апаратних зв’язків з останнім (рис. 4).

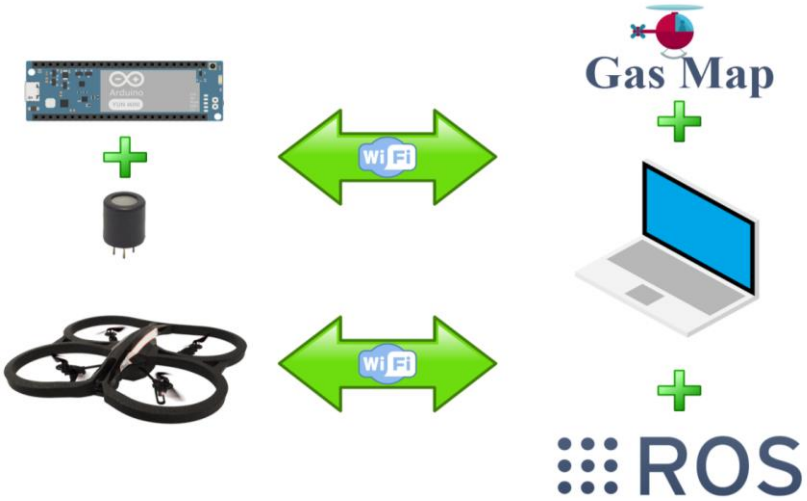

Рис. 4. Схема встановлення зв’язку між AR.Drone v2 та наземною станцією

Власноруч створений пакет ROS включає в себе абонента і слухача (відповідно до структури фреймворку). Ці два компоненти необхідні для інтеграції та надання доступу до даних вимірювань в ROS середовищі. Раніше в цьому розділі було описано, як саме працюють ці компоненти. Тепер цей робочий простір можна використовувати як в режимі моделювання так і в режимі реального часу. Для моделювання траєкторії польоту необхідним є режим моделювання. Наприклад, якщо вперше необхідно зробити деякі вимірювання на місці з попередньо невідомим середовищем, можна створити та використати недеталізовану модель місця польоту. Це дозволяє перевірити траєкторію польоту на предмет можливих колізій, оскільки вони можуть вивести з ладу БПЛА з вимірювальною системою на тривалий час, поточний стан проекту не включає розробку системи запобігання зіткнень. Відповідно до зазначених факторів потребується ретельна підготовки перед польотом, особливо в місцях з попередньо невідомим оточенням. Цей режим не потребує фізичної наявності БПЛА, оскільки являє собою лише віртуальне моделювання польоту БПЛА. Крім того симуляція допоможе створити маршрут польоту для дослідження:

Етапи підготовки до реального польоту:

– запуск tum_simulator ROS пакунку (рис. 5) [11];

– запуск tum_ardrone ROS пакунку і перемикання до ручного режиму керування;

– після цього пролетіти за певним маршрутом з увімкненим записом даних одометрії (політ відбувається в замкнутій симуляції – рис. 5);

– після виконання попередніх пунктів можливо перезапустити симуляцію і відтворити раніше записані дані одометрії, і відповідно до цього може бути створений та відкоригований план автоматичного польоту для відпрацювання пакетом tum_ardrone;

– gотім імітуйте політ зі створеним планом польоту і у випадку, якщо отримані результати задовольняють поставлені задачі, то наступним кроком повинен стати справжній політ.

Режим справжнього польоту. Для цього режиму не потрібно використовувати пакет моделювання, а потрібно запустити спеціальний пакет – test1, він включає служби видавця. Після запуску служби видавця можна запустити утиліту rqt_plot ROS і зберегти дані вимірювання газу (якщо це необхідно). Абонент з пакунку test1 може бути використаний в рамках інших служб для отримання вимір рівня концентрації з датчику газу, встановленому на квадрокоптері, в режимі реального часу. Цей режим має два підрежими: перший – реального часу, а другий – автономний.

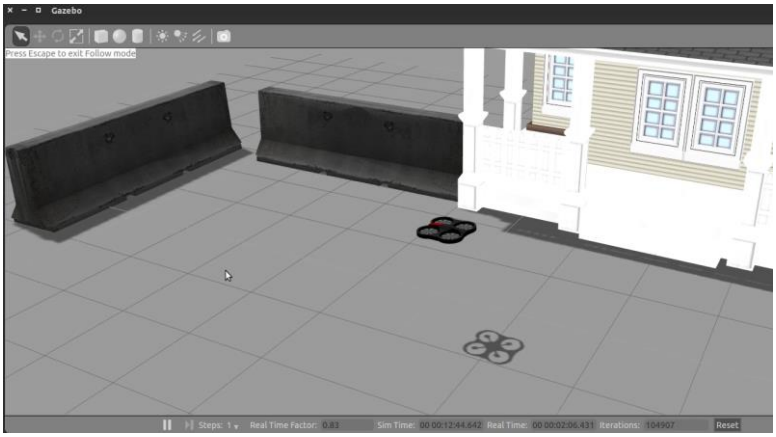

Рис. 5. Головне вікно tum_simulator ROS пакету

Запис льотної інформації. У процесі створення газової карти можна записати дані польотної інформації та вимірювання газу. AR.Drone v2 записує дані наступним способом: програмний додаток для газової мапи має можливість зберегти всі важливі дані в спеціальному файлі GeoJSON [12]. Геометричний масив включає x, y, z декартові координати AR.Drone Parrot v2. Також до масиву координат включено географічну довготу і широту (це необхідно для візуалізації газових точок на мапі). Marker_color у масиві властивостей містить 4 числа у форматі шістнадцяткового об'єднання, по 2 символи на число (для прикладу: ff або 8A), в одному рядку, і це описує колір RGBA, який формується відповідно до значень вимірювань підключених датчиків.

Для запису даних польоту використовувалася команда, подібна до цієї:

rosbag record -O ~/bagfiles/flight1.bag /ardrone/odometry /ardrone/image_raw, де rosbag – назва пакета ROS для запису і відтворення попередньо записаних даних;

record – перший параметр для цієї команди, це означає, що тепер інформація записується;

аргумент -O наказує rosbag записувати журнал у файл з ім'ям flight1.bag, з розташуванням у каталозі ~/ bagfile, а службові аргументи дають команду на запис rsbag тільки підписки на ці дві теми (/ardrone/odometry та /ardrone/image_raw);

/ardrone/odometry – ROS служба що містить дані одометрії;

/ardrone/image_raw – містить відео з камер AR.Drone Parrot.

Запуск цієї команди виконує конвертацію flight1.bag до вигляду звичайного .txt файлу логів з одометричною інформацією:

rostopic echo -b ~/bagfiles/flight1.bag -p /ardrone/odometry > data.txt

Звичайно, ця команда може бути модифікована для виокремлення відео з flight1.bag і його послідуючого перетворення в формат звичайного відеофайлу.

Після отримання файлу .bag і файлу даних газу, а також перетворення файлу .bag в файл odometry.txt можна побудувати карту газу.

Побудова мапи газу. Загальна схема зв'язку між дроном і наземною станцією описується на рис. 6. Плата Arduino підключається до термодатчика датчика, у зв'язку з неможливістю проведення повноцінних льотних випробувань поблизу місця значного витоку газу датчик вимірювання концентрації газу було замінено на термодатчик, і розміщується на борту AR.Drone Parrot v2 (у випадку використання попередньої платформи HW AR.Drone v1 підключається до газового датчика).

Загалом управління AR.Drone здійснюється через 3 основні служби зв'язку. Управління та налаштування дрону здійснюється шляхом відправлення команд AT на UDP-порт 5556. Інформація про безпілотник (наприклад, його статус, його положення, швидкість, швидкість обертання двигунів тощо), що називається navdata, надсилається дроном своєму клієнту на порт UDP 5554. Відеопотік відправляється AR.Drone на клієнтський пристрій на порт 5555 (UDP використовується для AR.Drone 1.0, TCP для AR.Drone 2.0). Четвертий канал зв'язку, що називається керуючим портом, може бути встановлений на TCP-порту 5559 для передачі критичних даних [13].

Arduino Yun створює власне TCP-з'єднання на порту 255. Наземний комп'ютер створює точку доступу Wi-Fi. Arduino Yun і AR.Drone v2 підключаються до цієї точки доступу Wi-Fi (рис. 4).

Для побудови карти газу використовується спеціальна програма (джерельний код цієї програми доступний у відкритому репозиторії github [14]).

На рис. 6 зображено діаграму програмної реалізації. На цій діаграмі є два сірі блоки. Сірі блоки – це зовнішні компоненти. Блок Files включає 2 файли: Geo File – вміст даного файлу, відправляється до Google Maps API, що дозволяє відобразити поточну позицію квадрокоптеру на Google Maps, Json File – це файл для збереження інформації про поточний стан вимірювань. Створення файлу Json записує поточний політ для подальшого відтворення в автономному режимі. Іншим сірим блоком є вхідні компоненти (Input Components). Вхідними компонентами є Arduino (підключена за допомогою TCP) і ROS (вузли, які надають інформацію про поточне положення квадрокоптера і поточні часові мітки даних).

Поруч із сірими блоками розташовані три кольорові блоки. Ці блоки є реалізацією шаблону програмування MVC. Згідно з цим шаблоном програма повинна складатися з моделей (Models), відображення (Views) і контролерів (Controllers).

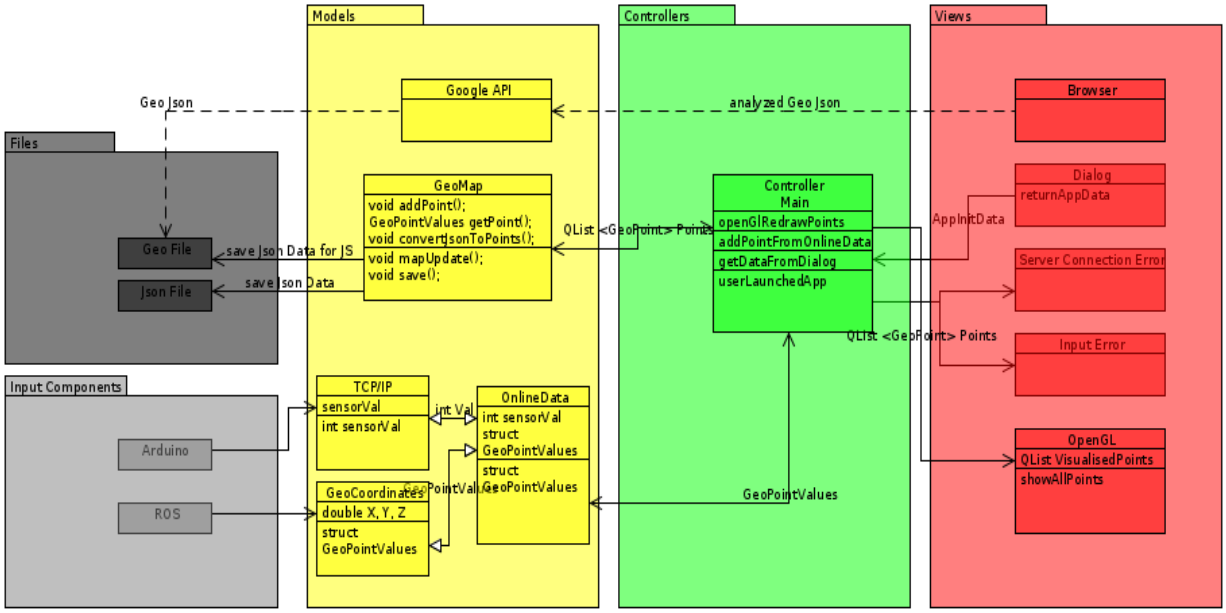

Рис. 6. Діаграма програмної реалізації

Всі класи, що належать до моделі, працюють з даними програми. Клас Google Map включає в себе всі методи, які відповідають за збереження даних про точки вимірювання і операції збереження даних координат польоту в Json File і відновлення хмари точок з цього файлу в разі запуску програми в автономному режимі. Online data – обгортка для класів TCP/IP і GeoCoordinates. Клас TCP/IP відповідно до свого імені відповідає за з'єднання по TCP/IP протоколу між платою Arduino Yun та програмою побудови мапи газу, встановленою на ПК наземної станції. Це необхідно для отримання даних датчиків, які підключені до Arduino Yun. Клас GeoCoordinates отримує дані одометрії від ROS і здійснює певні трансформації цих даних (наприклад, перетворення координат декартових в координати географічного розташування). Google API буде описуватися самостійно в складі компонента Browser (View).

Класи відображення (View) надають можливість візуалізувати інформацію представлену в моделі для користувача і вони необхідні для зв'язку між користувальницьким інтерфейсом і моделлю (рис. 7, рис 8). Найбільш важливим з цих класів є клас OpenGL. Клас OpenGL створює Open GL [15] вікно, що візуалізує хмару точок газових вимірів. Клас Dialog надає можливість встановлення параметрів ініціалізації, як онлайн/офлайн режим, мінімальне/максимальне значення датчика (програмний діапазон чутливості підключеного датчика), початкове місце знаходження дрону (поточна система не має GPS-модуля, тому гео-координати – це декартові координати, перетворені в географічні координати) або ім'я файлу без згаданих раніше параметрів у випадку автономного режиму. Помилка підключення до сервера (Server Connection Error) – це вікно з помилкою, яке з'являється у випадку вибору онлайн-режиму без можливості підключення до платформи AR.Drone Parrot. Помилка вводу (Input Error) буде показано у випадку помилкових даних у будь-якому полі вікна форми діалогу.

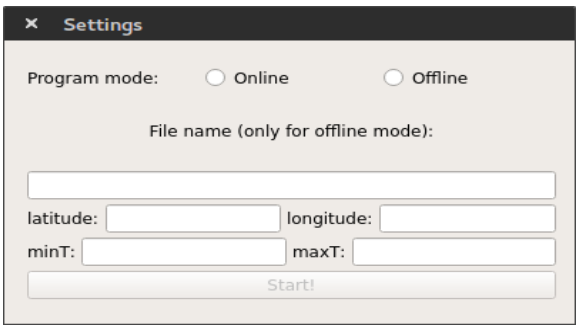

Рис. 7. Вікно початкової конфігурації програми

Головний контролер (Main Controller) – це компонент, що забезпечує можливість зв'язку між частинами моделі та відображення.

Google API + Browser відповідають за функціональність карт Google (вона показує точки вимірювання газу на картах Google). Насправді можна інтегрувати браузер в головний додаток як інше вікно QT, але використання незалежного браузера (для запуску JS скрипту) краще, тому що він може працювати на іншому (інших) ПК для відслідковування поточного розташування БПЛА. Більш того, з міркувань безпечної розробки програмного забезпечення краще зробити його відокремленим (рис. 9).

Використання MVC паттерну програмування дає можливість легкого масштабування програмного продукту і можливість додавання додаткових функцій без необхідності внесення зміни в раніше написаний код.

В основному дане програмне забезпечення, написане на мові програмування C ++ з використанням QT5 фреймворку, але також включає скрипти запуску bash, скрипт python в частині ROS і скрипт JavaScript для використання API Google Maps (Google не надає C++ API, вони пропонують лише JS API).

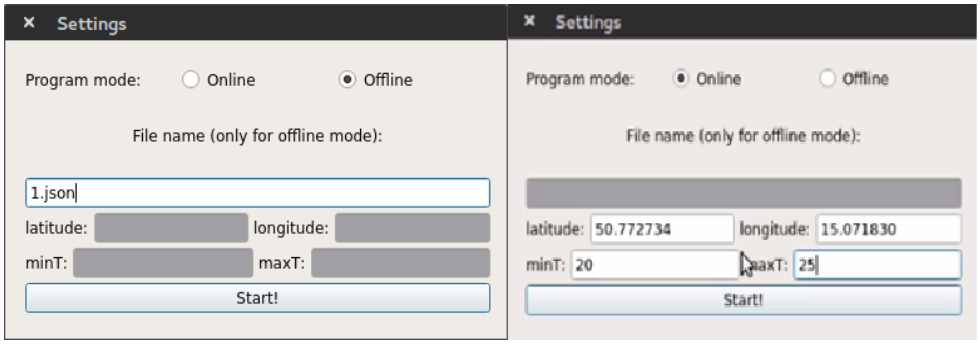

Рис. 8. Вікно початкової конфігурації програми в автономному та режимі реального часу

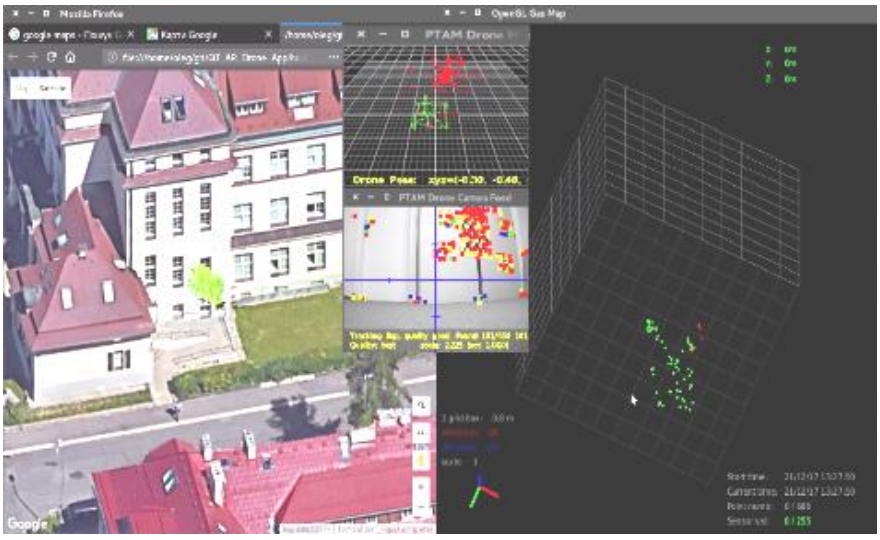

Рис. 9. Головне вікно розробленої програми для побудови «карти газу» (праворуч), браузер з запущеним JS скриптом, який відображає маршрут проведення вимірів на місцевості (ліворуч), Drone Camera та Drone PTAM вікна з tum_ardrone ROS пакунку (всередині)

Програма зображує газові хмари в 3D-просторі. Вона надає можливість обертати попередньо побудовану 3D-сцену, щоб обрати кращу позицію для перегляду газової хмари, для обертання навколо осі потрібно натиснути ліву клавішу мишки і перемістити стрілку на екрані це дозволить змінювати напрямок огляду навколо осі X та Z. Перш ніж використовувати цю програму, потрібно налаштувати такі постійні значення, як масштаб карти початкові координати точки запуску (в проекті не використовується GPS тому географічні координати отримуються математичним шляхом). Щоб переглянути дані однієї з візуалізованих точок, необхідно натискати клавіші "+" або "-" на клавіатурі доти поки не буде обрана потрібна точка на екрані. Також є рядок стану, який візуалізує кількість точок у відсотках. Рівень газу описується кольором точки. Він змінюється від зеленого до червоного з жовтим у середині відповідно до кількості газу в певний момент часу в діапазоні вимірювань.

Якщо програма використовується в режимі реального часу допоміжні компоненти будуть запущені автоматичним скриптом. Основною відмінністю між програмами є інший алгоритм отримання даних. В режимі реального часу програма повинна підключитися через wi-fi до квадрокоптера. У режимі offline програма відновить одну з попередніх сесій, що була автоматично збережена під час одного з раніше проведених експериментів.

Стабільність роботи в режимі реального часу забезпечується перевіркою якості даних газу та урахуванням лише реальних даних, також використання протоколу TCP/IP гарантує відповідність отриманих даних реальним вимірам навіть у місцях з низьким рівнем сигналу WiFi.

Алгоритмічне забезпечення. На рис. 10 представлено основний алгоритм автоматизованої мобільного бездротової системи моніторингу якості повітря.

Результати дослідження

Випробування програми побудови мапи газу. Як вже згадувалося раніше, було використано датчик SHT10 для тестування програми побудови мапи газу замість датчика концентрації газу. Тести з газом потребують особливих умов, тому тестування програмних компонентів виконувалося в умовах відсутності додаткового газу. Рис. 11 демонструє умови проведення експерименту. Нагрівач (чорний ящик під AR.Drone Parrot) створює спрямований тепловий потік. Безпровідна вимірювальна система на базі AR.Drone Parrot v2 виконує політ відповідно до попередньо визначеного маршруту і проводить вимірювання температури під час цього польоту.

Результати цього тесту показані на рис. 12.

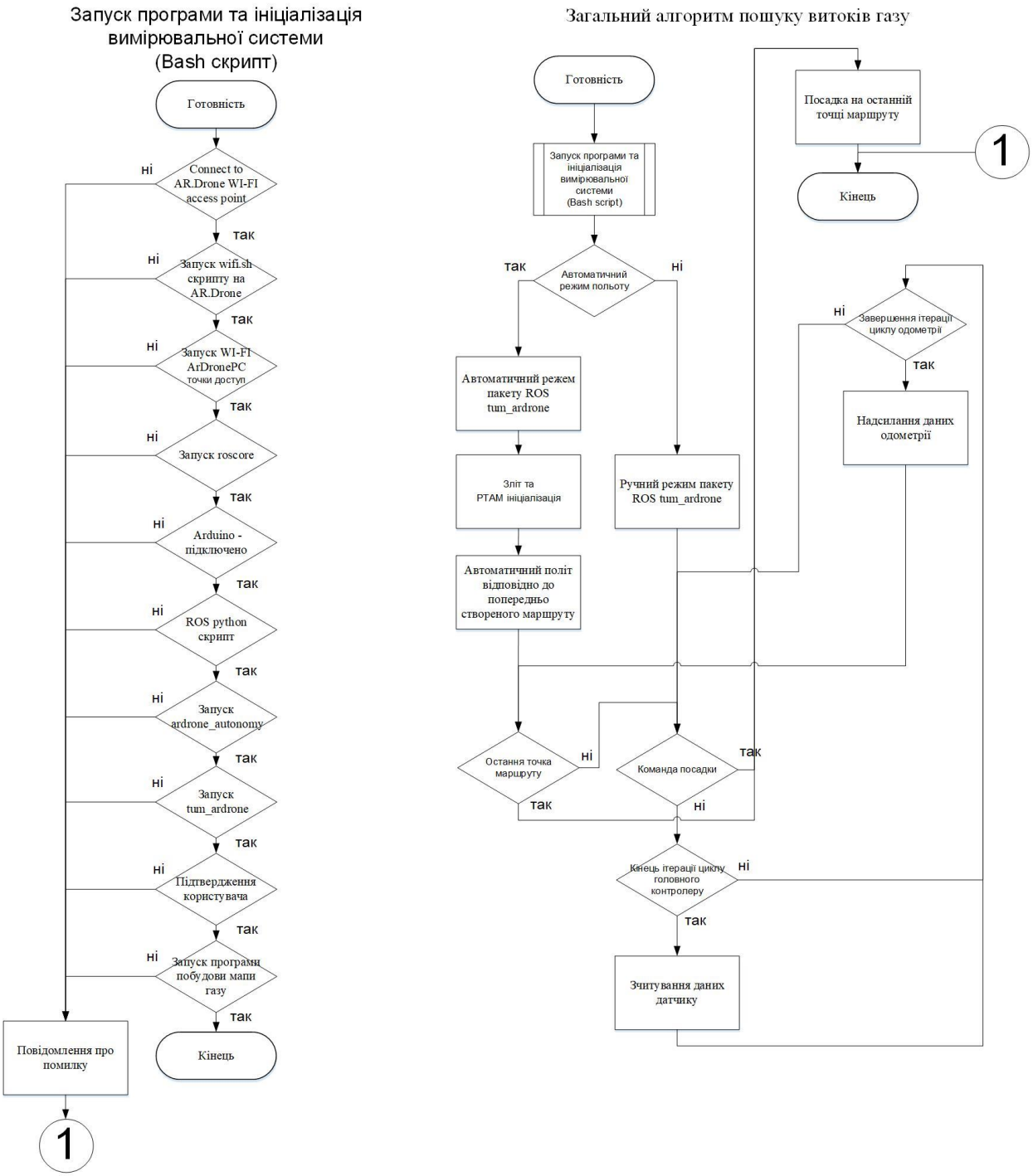

Рис. 10. Основний алгоритм автоматизованої мобільного бездротової системи моніторингу

Рис. 11. Тестова сцена під час процесу абстрактного виявлення газу

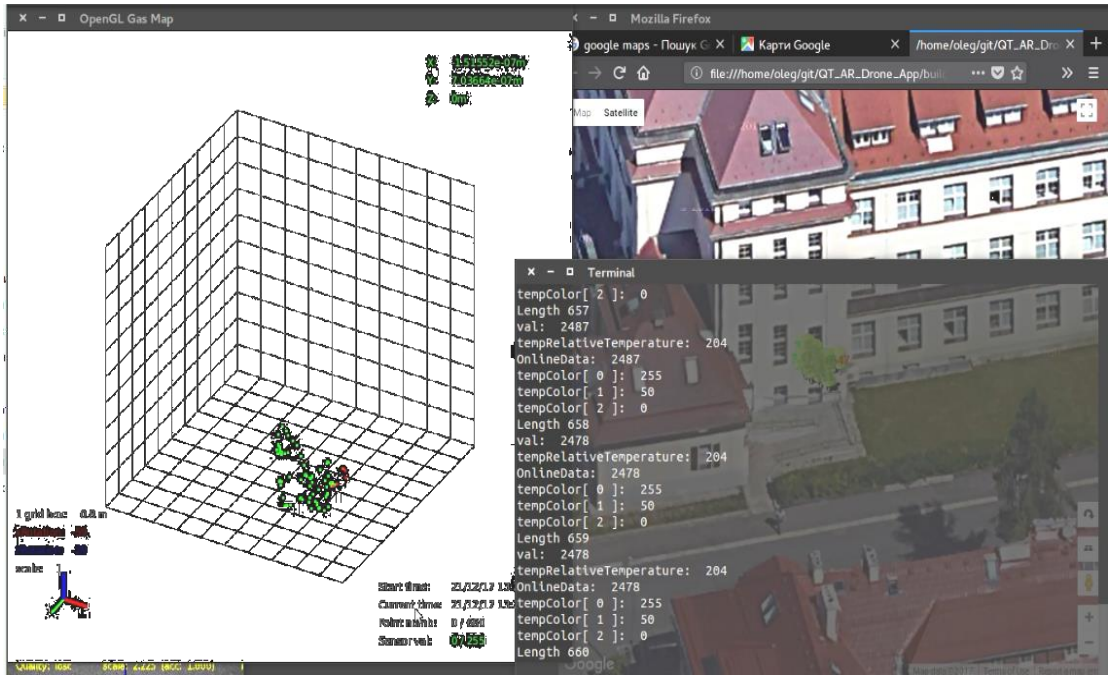

Рис. 12. Мапа газу під час тестування програми побудови мапи газу

Висновки

У ході даного проекту було розроблено мобільну систему для моніторингу якості повітря та наявності в ньому небезпечних газів на основі БПЛА. В ході розробки та тестування даної системи було створено програмний продукт для дистанційного отримання даних та їх обробки з подальшим складанням мапи загазованості території для візуалізації отриманих даних. Система має зв'язок з наземною станцією для можливості отримання результатів в режимі реального часу та їх миттєвої обробки.

References

1. Vachtsevanos G. J., Valavanis K. P. (2014) Handbook of Unmanned Aerial Vehicles. (Springer Publishing Company).

2. Koval A., Irigoyen E. (2017) Mobile Wireless System for Outdoor Air Quality Monitoring. In: Graña M., López-Guede J., Etxaniz O., Herrero Á., Quintián H., Corchado E. (eds) International Joint Conference SOCO'16-CISIS'16-ICEUTE'16. ICEUTE 2016, SOCO 2016, CISIS 2016. Advances in Intelligent Systems and Computing, vol 527. Springer, Cham.

3. Quigley M., Conley K., Gerkey B., Faust J., Foote T., Leibs J., Wheeler R., Ng A.Y. (2009) Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software, vol. 3, p. 5.

4. Mani Monajjemi. Ardrone autonomy. Mani Monajjemi (2015). URL: https://ardrone-autonomy.readthedocs.io/en/latest/installation.html.

5. Repository for the tum_ardrone ROS package, implementing autonomous flight with PTAM-based visual navigation for the Parrot AR. Drone (2014). URL: http://wiki.ros.org/tum_ardrone.

6. Adam A. How to Calibrate a Monocular Camera (2017). URL: http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration.

7. Grey C. Tutorial for calibrating the camera (2013). URL: http://wiki.ros.org/ethzasl_ptam/Tutorials/camera_calibration.

8. Arduino Yún. 2017. URL: https://store.arduino.cc/arduino-yun.

9. Sensor Humidity SHT1x datasheet (2010). URL: Rethttps://cdn.sparkfun.com/datasheets/Sensors/Pressure/Sensirion_Humidity_SHT1x_Datasheet_V5.pdf.

10. ROS Tutorials (2017). URL: http://wiki.ros.org/ROS/Tutorials.

11. Hongrong H., Jürgen S. Tum_simulator (2014). URL: http://wiki.ros.org/tum_simulator.

12. GeoJSON (2016). URL: http://geojson.org/.

13. AR.Drone Developer Guide (2012). URL: https://jpchanson.github.io/ARdrone/ParrotDevGuide.pdf.

14. Github with code of the project (2017). URL: https://github.com/OlehQWERTY/QT_AR_Drone_App.git.

15. OpenGL Overview. URL: https://www.opengl.org/about/

Рецензія/Peer review : 3.11.2019 р. Надрукована/Printed : 04.01.2020
Рецензент: д.т.н., проф. Полонський Л.Г.