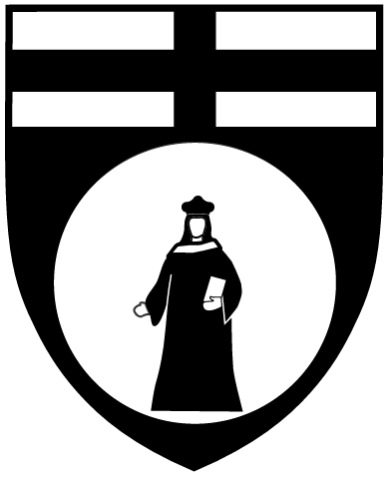

Dibris Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi

Virtual Reality

Report

Air pollution monitoring in a Smart City

Advisor: Gianni Viardo Vercelli

UNIVERSITY OF GENOA, SEPTEMBER 2023

Member list & Workload

No.	Full name	Student ID	Percentage of work
1	Aurora Durante	4647689	50%
2	Martina Germani	4632805	50%

Contacts

Aurora Durante (aurora.durante@coservizi.it)
Martina Germani (martinella711@gmail.com)

Both developpers are Master Students in Robotics Engineering in UNIGE, Genoa.

Contents

1 Introduction

A *Smart City* is defined as an urban area in which technologies are applied to better improve the efficiency of infrastructures and city services. In this way, a Smart City is supposed to be able to actively monitor and act on the environment to enhance its citizens quality of life.
To achieve this goal, technologies should be applied to control *pollution level* in the environment, which can be divided in air pollution, water pollution and soil pollution. For each type the focus of the monitoring is different and regarding air pollution it can be transportation, energy production, wastes management and indoor pollution.
In this way *drones* equipped with sensors for air pollutants as O_3, SO_2, NO_2, PM_{10} and $PM_{2,5}$ can be applied in this context.

In our project we aim at simulating and monitoring air pollution in a Smart City through the deployment of patrolling aerial drones.
We assumed the availability of a major metropolis, such as London, that we aimed to monitor for pollution. Our focus was on combining the efforts of a mobile unit, which will navigate the city streets, with a drone that will assess pollution levels in predetermined high-risk areas.
So, by combining the data collected from both the mobile unit and the drone, we will be able to generate a comprehensive pollution map of the city. This map will highlight areas with elevated pollution levels, enabling us to identify problematic areas and implement targeted measures to mitigate pollution.

For controlling both the ground vehicle and the drone, we implemented a *Finite State Machine* (FSM) in order to switch between the two game modes of AirSim on Unreal Engine simulation.
The control of the drone in the system is based on another group project running on Robot Operating System (ROS) framework, which provides a powerful and flexible architecture for building robotic systems, and AirSim ROS Wrapper, which creates a bridge between Unreal Engine and ROS so that drones can be controlled and visualized in the engine.
Instead, as approach for simulating changes in the pollution level, we created with Blueprint on Unreal Engine a block of **empty** type which divides the air above the areas of interest (e.g areas characterized by traffic in certain time slots) in cubes/rectangles and assign them a *pollution* variable.

Finally, once the drones acquired all the data, we calculated the *Air Quality Health Index* (AQHI) as the *World Health Organization* requires to give the citizens information on how to better behave consequently.

Here is the link to our project GitHub repository with all the information needed for installing and running it:

https://github.com/AuroraD-Hub/VR_Assignment

2 State of the Art

In recent years, the use of Unmanned Aerial Vehicles (UAVs), commonly known as drones, has gained significant attention in various fields, including environmental monitoring. In particular, UAVs have proven to be valuable tools for monitoring pollution levels and assessing environmental conditions.

In this chapter we will explore the application of UAVs in the field of pollution monitoring and highlight their advantages and disadvantages, challenges, and potential for enhancing our understanding of environmental pollution.

2.1 UAVs for Pollution Monitoring

UAVs have revolutionized the field of data gathering and pollution monitoring by providing a versatile platform that can navigate challenging terrains, reach remote locations, and capture high-resolution images and sensor readings. These capabilities have made UAVs increasingly popular for monitoring air quality, water pollution, and industrial emissions.

Several studies have investigated the application of drones in this area, focusing on various aspects of pollution detection, mapping, and forecasting. These studies have explored the capabilities of drones in measuring and monitoring different pollutants, including SO_2, NO_2, particulate matter ($PM_{2.5}$ and PM_{10} specifically), volatile organic compounds (VOCs), ozone (O_3), and water vapor ([2], [3], [12]).

Researchers have developed innovative systems that combine UAV-captured imagery with ground-based sensor networks to perform fine-grained air quality monitoring. This approach allows for the accurate assessment and prediction of air quality conditions in spatial-temporal perspectives. By utilizing multi-sensor data fusion techniques, drones can provide a comprehensive and detailed understanding of pollution sources, dispersion patterns, and the overall air quality landscape [18].

Moreover, the integration of solar-powered UAVs into wireless sensor network (WSN) systems has addressed the issue of limited power consumption, thereby extending flight time and enhancing monitoring capabilities. These solar-powered UAVs have been employed to continuously monitor greenhouse gases, taking advantage of renewable energy sources and overcoming the constraints associated with traditional power systems [6].

The versatility and maneuverability of drones enable them to access challenging environments, such as industrial areas or remote locations, where pollution sources may be concentrated. Drones equipped with specialized sensors can collect real-time data, offering a comprehensive picture of pollution hotspots and aiding in the identification of potential health and environmental risks.

The successful applications of drones in environmental air pollution monitoring have paved the way for the development of autonomous Environmental Drones (E-drones) [9]. These programmed drones are specifically

designed for pollution monitoring, detection, and abatement. E-drones produce Air Quality Health Index (AQHI) maps [15], enabling the monitoring and long-term analysis of environmental data. They represent a significant advancement, as they combine pollution detection with the ability to take immediate action for pollution abatement. These actions follow air quality and health directives from the *World Health Organization* [8] which consider to calculate the AQHI with the NowCast system introduced by US EPA [17] [14]. To understand the formula, consider for each pollutant: N as the number of hours of the monitoring period and c_i as the sampled values at hour i. Here as it is calculated:

$$w^* = \frac{c_{min}}{c_{max}}$$

$$w = \begin{cases} w^*, & \text{if } w^* > \frac{1}{2} \\ \frac{1}{2}, & \text{otherwise} \end{cases} \tag{1}$$

$$AQHI^{pol}_{NowCast} = \frac{\sum_{n=1}^{N} w^{i-1} c_i}{\sum_{n=1}^{N} w^{i-1}}$$

Moreover, there are different ways to define the overall index and all are based on National directives. In the specific case of London [16], once the AQHI is obtained for each pollutant by following the table in Figure 1, the overall index is defined as the maximum among them as recommended by the *Committee on the Medical Effects of Air Pollutants* (COMEAP) of the United Kingdom. This index is based on a 10-point scale which is then divided into four bands highlighting the level of pollution and corresponding level of danger as can be seen in the table in Figure 2.

Index	Ozone, running 8 hourly mean (µg/m^3)	Nitrogen dioxide, hourly mean (µg/m^3)	Sulphur dioxide, 15 minute mean (µg/m^3)	$PM_{2.5}$ particles, 24 hour mean (µg/m^3)	PM_{10} particles, 24 hour mean (µg/m^3)
1	0–33	0–67	0–88	0–11	0–16
2	34–66	68–134	89–177	12–23	17–33
3	67–100	135–200	178–266	24–35	34–50
4	101–120	201–267	267–354	36–41	51–58
5	121–140	268–334	355–443	42–47	59–66
6	141–160	335–400	444–532	48–53	67–75
7	161–187	401–467	533–710	54–58	76–83
8	188–213	468–534	711–887	59–64	84–91
9	214–240	535–600	888–1064	65–70	92–100
10	≥ 241	≥ 601	≥ 1065	≥ 71	≥ 101

Figure 1: Air Quality Health Index for each pollutant

Air pollution banding	Value	Health messages for at-risk individuals	Health messages for general population
Low	1–3	Enjoy your usual outdoor activities.	Enjoy your usual outdoor activities.
Moderate	4–6	Adults and children with lung problems, and adults with heart problems, who experience symptoms, should consider reducing strenuous physical activity, particularly outdoors.	Enjoy your usual outdoor activities.
High	7–9	Adults and children with lung problems, and adults with heart problems, should reduce strenuous physical exertion, particularly outdoors, and particularly if they experience symptoms. People with asthma may find they need to use their reliever inhaler more often. Older people should also reduce physical exertion.	Anyone experiencing discomfort such as sore eyes, cough or sore throat should consider reducing activity, particularly outdoors.
Very High	10	Adults and children with lung problems, adults with heart problems, and older people, should avoid strenuous physical activity. People with asthma may find they need to use their reliever inhaler more often.	Reduce physical exertion, particularly outdoors, especially if you experience symptoms such as cough or sore throat.

Figure 2: Air Quality Health Index table used in UK

2.2 Advantages

The utilization of drones for pollution monitoring offers several significant advantages. These advantages stem from the unique capabilities and characteristics of drones, making them a valuable tool in this field.

First and foremost, drones provide unparalleled mobility and flexibility. With their ability to navigate challenging terrains and reach remote or inaccessible locations, drones can access areas that are difficult for humans to reach. This enables comprehensive monitoring of pollution in diverse environments, including industrial zones, urban areas, and natural ecosystems.

Moreover, drones offer a cost-effective solution compared to traditional monitoring methods. Traditional methods often involve manual data collection, which can be time-consuming and labor-intensive. Drones, on the other hand, can cover large areas efficiently and collect data in real-time, reducing both time and human resource requirements. This cost-effectiveness makes it feasible to conduct more frequent and widespread pollution monitoring, leading to a better understanding of pollution patterns.

Another advantage of using drones is their ability to provide high-resolution images and sensor readings. Equipped with advanced sensors, drones can capture detailed and accurate data on various pollutants, such as particulate matter, greenhouse gases, and volatile organic compounds. This high-resolution data enhances the precision and reliability of pollution assessments, facilitating targeted interventions and informed decision-making.

Drones also offer the potential for real-time data collection and analysis. The data collected by drones can be transmitted and processed in real-time, allowing for immediate evaluation and response. This capability enables rapid identification of pollution hotspots and prompt implementation of measures to mitigate pollution sources, ensuring a more timely and effective environmental management approach.

Furthermore, the integration of drones with other technologies, such as remote sensing and data fusion techniques, enhances the capabilities of pollution monitoring systems. Drones can be used in conjunction with satellite imagery, ground-based monitoring stations, and modeling tools to provide a comprehensive and holistic understanding of pollution sources, transport patterns, and their impact on the environment.

2.3 Disadvantages

While drones offer the many advantages just discussed for monitoring pollution, it is important to recognize some potential disadvantages. One significant limitation is the limited flight time and range of drones due to their reliance on batteries. This constraint requires frequent battery changes or the deployment of multiple drones for large-scale monitoring, resulting in higher costs and logistical complexity.

Another consideration is the dependence of drones on favorable weather conditions. Adverse weather conditions, such as high winds, rain or extreme temperatures, can hamper drone flights and data collection. These weather-related limitations can lead to data gaps or delays in acquiring essential information.

Drones also have payload limitations, limiting the types of sensors and equipment they can carry. Weight limitations can affect the accuracy and precision of your measurements, particularly when dealing with heavy or complex monitoring tools.

Processing and analyzing the large volumes of data collected by drones presents a computational challenge. Advanced data processing techniques are required to extract meaningful information, which can be time consuming and resource intensive. Appropriate skills and resources are required for efficient management and analysis of data captured by drones.

Regulatory and legal considerations are additional factors to address. Compliance with aviation regulations and privacy laws is essential, requesting permits and ensuring data confidentiality. Complying with these requirements brings administrative complexity and potential additional costs.

Safety concerns also arise when using drones in shared airspace with manned aircraft. Collisions or accidents carry risks, especially in densely populated or heavily trafficked areas. Implementing safety protocols, training drone operators and adhering to aviation guidelines are crucial steps in mitigating these risks.

Public acceptance and perception of drones can also be a challenge. Privacy invasion and security risks are common concerns raised by the public. Transparent data collection practices, community engagement, and addressing privacy concerns are key to fostering public acceptance.

Recognizing and addressing these potential disadvantages is critical to the successful implementation of drone-based pollution monitoring initiatives. By mitigating these challenges, the benefits of using drones can be maximized, enabling more effective and sustainable environmental monitoring practices.

2.4 Challenges

In conclusion, the use of drones for pollution monitoring presents certain challenges that need to be addressed. The limited flight autonomy, dependency on weather conditions, and payload restrictions can impact the scope and effectiveness of drone operations. Processing and analyzing the collected data require specialized resources and expertise. Furthermore, ensuring regulatory compliance, addressing privacy concerns, and fostering public acceptance are crucial aspects. Effectively tackling these challenges will maximize the potential of drones in pollution monitoring, leading to a better understanding and management of pollution phenomena to safeguard the health of our planet and communities.

3 Tools

For this project we created a simulated environment with *Unreal Engine* (version 4.27.2) [5] and used *AirSim* [7] and ROS [10] to control the patrolling drones. The user interface managing the deployment of the car to move through the city or the drones to air sample is based on a FSM [11] in Python.

3.1 UE environment

To create a simulated city environment for our project, we searched in the Marketplace and found the *AccuCities Textured Sample* by AccuCities. This particular resource offers a highly detailed 3D replica of a 1 square kilometer area of London.

AccuCities is renowned for its expertise in creating accurate and realistic urban models. Their textured sample provides a comprehensive and immersive representation of a cityscape, capturing the architectural details, landmarks, and spatial layout of London. By utilizing this textured sample, we can effectively simulate the environmental conditions of a bustling metropolis like London, which serves as an ideal backdrop for our pollution monitoring project.

In this package the are two maps: *AccuCities_ Separate_ TextureModel* and *AccuCities_ Unioned_ TextureModel*. We used the latter one because the *Spearate* version had collision bugs when the simulation started in *AirSimGameMode*.

3.2 AirSim

We used AirSim with Unreal Engine for controlling both a ground vehicle and a drone in our project. It is an open-source simulation platform for testing autonomous systems, specifically drones, in a virtual environment. It provides a high-fidelity simulation environment that accurately models the physics and dynamics of aerial vehicles, as well as a variety of sensor models for testing perception and control algorithms. AirSim is a plugin for Unreal Engine, providing a powerful and flexible platform for developing and testing autonomous systems in a safe and controlled environment. Its potentialities are enhanced by the development of a ROS Wrapper and an API both in C++ and in Python.

3.2.1 AirSim ROS Wrapper

In our research, we explored the integration of the Robot Operating System (ROS) with AirSim to enhance our pollution monitoring capabilities. ROS, a widely adopted open-source framework in robotics, provides a flexible and modular platform for controlling autonomous systems. By combining ROS with AirSim, a realistic and immersive simulator for autonomous vehicles, we created a connection between the simulated environment and our control algorithms. This integration offers several advantages, including access to ROS extensive library

of tools and algorithms, facilitating the development and testing of our control strategies. Additionally, ROS distributed architecture enables concurrent and collaborative development, enhancing modularity and code reusability. The integration also allows for easy transition from simulation to real-world deployment, reducing risks and costs. Furthermore, the ROS-AirSim integration enables data exchange between simulated vehicles and real-world sensors, facilitating the validation and refinement of perception algorithms.

3.2.2 AirSim API

The developed API allows to interact programmatically with the spawned vehicle in the simulation.
It creates a client and provides functions to control the vehicle and to get its state, but also methods to retrieve images from cameras, pause the simulation, change time-of-day and weather settings, record the simulation, control multiples vehicles and so on. Since AirSim offers the possibility to play in two GameModes (`Car` and `Multirotor`), there are also specific instances of the client that can be used: *CarClient* and *MultirotorClient*.
Finally, the API is implemented in C++ since AirSim is a Plugin for UE that creates games and simulations in C++ or Blueprint, but there is also the availability of a Python version of the API.

3.3 Drone coverage controller

To sample the air in different heights, we used the controller implemented by `Dronati` group and adapted it to our purpose. It spawns through AirSim ROS Wrapper the drones specified in `setting.json` file and creates a controller for them such that they can follow a given path without colliding with each other and get back to base. More information can be found in their GitHub repository here:

https://github.com/mmatteo-hub/VR4R_Assignment.

3.4 SMACH ROS

SMACH is a Python library that can be used to create hierarchical state-machines. It is actually ROS-independent, but the `smach_ros` package is available so that ROS can be integrated with SMACH to build complex robot behaviours in a robust, modular and of easy maintainability way.
Since SMACH is a task-level architecture and it performs optimally when all possible states and state transitions can be described explicitly, we used it to change the GameMode in the simulation in accordance with AirSim API. In fact, our intent is to use the car to move from one location to another of the city while the drone to perform the pollution monitoring procedure.

4 Description

In this section we will describe firstly the custom Blueprint asset *Pollution* that we created to simulate the pollution level in specific areas of the city based on pollutants O_3, NO_2, SO_2, $PM_{2.5}$ and PM_{10}. Then, the structure and the code of our project is illustrated.

4.1 BP Asset Pollution

To simulate the air-pollution, we built a *StaticMeshActor* of the Blueprint class, which we called *Pollution* and which is visible in the *Content Browser* menu. It consists of an object within the game world that uses a static mesh to define its shape and visual appearance. This type of object can be placed, rotated and scaled within the game world, but will not undergo any intrinsic animation.
On this *StaticMeshActor* object we implemented an ***OnComponentBeginOverlap*** event, which is triggered when the collision component of the object comes into contact with other objects in the game world, in our case with the drone/car being piloted.
Therefore, every time our vehicle crosses one of these objects, the overlap event is triggered which activates the generation of the pollution values detectable at that point, which are then saved in *.../SavedData/PollutionData* file (Figure 3 shows the event graph of this Blueprint actor).

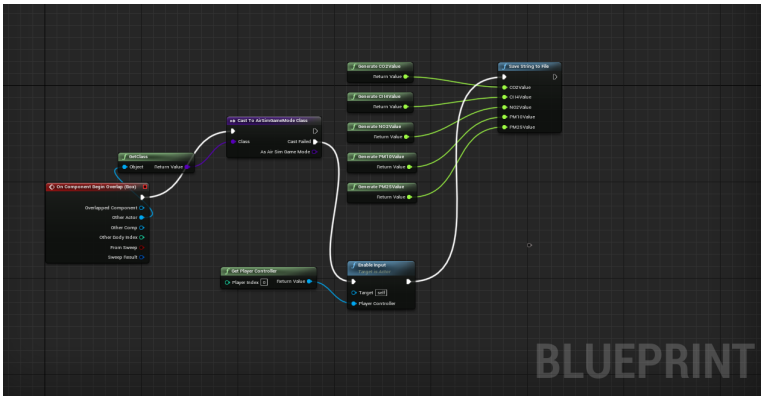

Figure 3: Event Graph of the Pollution StaticMeshActor

As you can see in Figure 3, which shows the event graph of this Blueprint actor, we created the following functions which derive from the *BlueprintFunctionLibrary* class so they can be accessed within Blueprint in the game engine:

- ***Save String To File***: it handles writing pollution data to a JSON file. If the file already exists, it adds new data to the existing array; otherwise, it creates a new JSON file with the provided data. This approach allows tracking data to be kept and updated on a persistent basis for any display, analysis or monitoring purposes;

- ***GenerateSO2Value***:it generates and returns a random SO2 (sulfur dioxide) value between 0 and 1215 µg/m^3, which represents the concentration of sulfur dioxide in the air;
- ***GenerateO3Value***:it generates and returns a random O3 (ozone) value between 0 and 270 µg/m^3, which represents the concentration of ozone in the air;
- ***GenerateNO2Value***: it generates and returns a random NO2 (nitrogen dioxide) value between 0 and 670 µg/m^3, which represents the concentration of nitrogen dioxide in the air;
- ***GeneratePM10Value***: it generates and returns a random value of PM10 (suspended particles with a diameter of less than 10 microns) between 0 and 110 µg/m^3, representing the concentration of PM10 particles in the air;
- ***GeneratePM25Value***: it generates and returns a random value of PM2.5 (suspended particles with a diameter of less than 2.5 microns) between 0 and 75 µg/m^3, representing the concentration of PM2.5 particles in the air.

The values generated by these functions refer to the table in Figure 2.
Therefore, every time a collision occurs, the JSON file is filled with a new section containing the pollution values detected. These values are then read by the drone/car, which updates the file itself with the position where the collision occurred. In this way, a map of the air pollution in the city is obtained.

4.2 Project code

First, in the following figure there is the overall architecture:

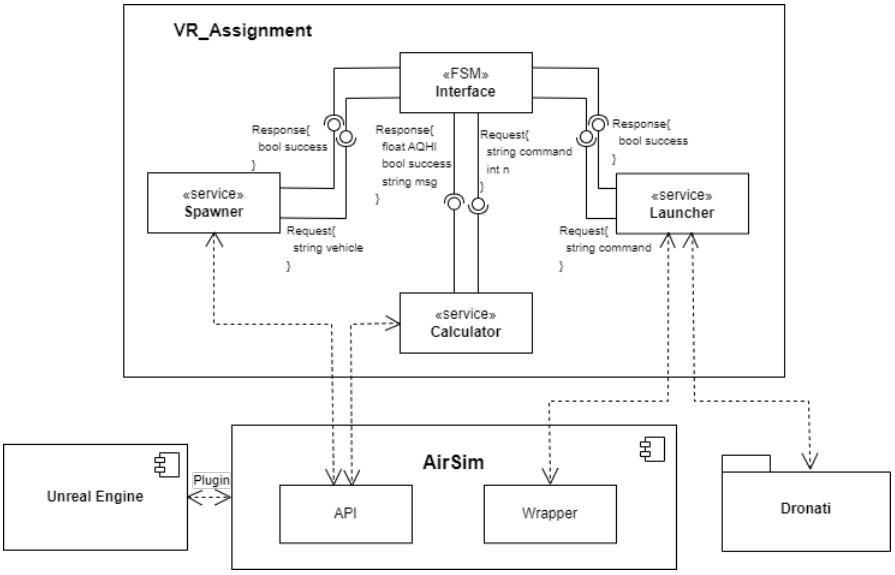

Figure 4: Project architecture

Now, each component of **VR_Assignment** is explained.

4.2.1 Interface

This simulation relies on a Finite State Machine (FSM) and the AirSim API to spawn and delete vehicles based on the specific scope of the vehicle. There are two states: *CarMoving* and *DroneFlying*.
It starts in *CarMoving* since AirSim is set up to Car SimMode and allows the user to control the vehicle with keyboard around the city to the chosen location in which the drone should sample the air. Once the location is reached, the user has to follow the instruction prompted on the terminal for the FSM to change state. In this way, the FMS calls a custom service `Spawner` that saves current position of the car, delete from simulation the vehicle and spawns a drone. At last, the FSM makes the transition to *DroneFlying*.
In this second state two gnome terminals are opened by another custom service `Launcher`: one for AirSim Wrapper and the other for the drone controller package. Follow the instruction on this last terminal to arm the drone and start the air sampling procedure managed by custom service `Calculator` so that a file containing information about the pollutants level and drone position can be updated. Once it is done, these gnome terminals are closed and the user just have to follow the instruction on the main one to make the transition to *CarMoving* again. Moreover, in *DroneFlying* the E-Drone is able to calculate the AQHI and give corresponding health messages for the civilians through the custom service `Calculator`.
In the following more detailed information about the custom services are given.

4.2.2 Custom service: Spawner

This service structure is composed by a string request (`vehicle`) and a Boolean response (`success`).
Based on the request, the service decides which AirSim client has to be used to delete the corresponding vehicle and this is done by executing **car_api** or **drone_api** method. In these methods, the functions **destroy_vehicle** and **spawn_new_vehicle** are called.
The former function manages to save the current position of the vehicle whichever it is thanks to the AirSim client, but with different methodology because of the dissimilar purpose of the vehicles. In fact, since the car is used to move the drone around the city, it is important that its last position is saved before deleting it so that a drone can be spawned in the same exact position. This is done by creating a JSON file with the same structure as in Listing 1, where fields `x`, `y` and `z` are compiled with car last position for every point `p_i` and where node `p_0` is located on the ground where the car was. Note that this file is also used by the drone to know which location it has to reach to sample the air and so z-position of node `p_i+1` is located 10m above the previous point and there are four nodes in total. After position is saved, the AirSim client is used again to delete the vehicle.

```
    {
    "center": "p_i",
    "nodes": {
        "p_i": {
          "position": [
            x,
            y,
            z
          ],
          "difficulty": 0
        },
    },
    "connections": [
        [
          "p_i",
          "p_i+1"
        ]
    ]
}
```

Listing 1: JSON drone graph

Then, a new vehicle has to be spawned with the latter function and this is done by the AirSim client.

4.2.3 Custom service: Launcher

This service structure is composed by a string request (`command`) and a Boolean response (`success`).
Based on the request, the service decides if terminals have to be opened or closed by using method `Popen` of **subprocess** class and this is done by executing **open_terminals** or **close_terminals**.
In the former two GNOME terminals are opened: one for the AirSim Wrapper and one for the Dronati Drone Controller. After following the instruction of the Controller shell by typing the name of the drone, the service also loads JSON file in Listing 1 for the Controller to create the optimal drone path by calling imported services `LoadCoverageGraph` and `ComputeCoveragePath`.
The latter function closes all the terminals previously opened by executing *pkill* on every subprocess.

4.2.4 Custom service: Calculator

This service structure is composed by a string and int request (`command` and `n` respectively) and a float, a Boolean and a string response (`AQHI`, `success` and `msg` respectively).

Based on the request, the service decides if the JSON file created by the custom Blueprint asset *Pollution* has to be updated or if the AQHI is required to be calculated and related health message to be displayed. This is done by executing `update_sampling_data` or `getAQHI` and `getHealthMessage` respectively.
The former updates the JSON file by adding current position of the vehicle that collides with the block to read pollution values and also a variable N, which define the time period required for computation of AQHI.
The function `getAQHI` calls two helper functions, `getNowCast` and `getPollutantIndex`, which respectively do the following: for every pollutant performs the NowCast algorithm in (1) to find out its pollution level and then define the corresponding AQHI for that pollutant based on Table 1. Finally, this function return the maximum among all the computed indices.
The function `getHealthMessage` returns the health message for at-risk population based on in which band defined in Figure 2 the calculated AQHI is.

5 Results

Unfortunately, this architecture couldn't work properly because of two main reasons: AirSim not allowing control of both vehicles in the same simulation and unadaptability of Dronati package to other projects.
Regarding the former, we found many difficulties in using AirSim for both car and drone especially when calling the Spawner to delete current vehicle and spawn the other. In fact, we managed to delete from the simulation at least visually the vehicle by calling the AirSim client function `simDestroyObject()`, but it seems the vehicle to still be present in the Garbage Collection since `listVehicles()` returns it as an object of simulation. For this reason whenever we ask the client of the other vehicle, that we manage successfully to connect with, to spawn the corresponding vehicle, UE crashes and closes the project. As an alternative method we tried to start the simulation with both vehicle already spawned and attached to move together, but then AirSim was still not able to control them mutually as we desired. In fact, the Wrapper couldn't manage the declaration of two different vehicles in `setting.json`.
Regarding the latter, we actually managed to integrate Dronati's Controller in our project successfully, but when it comes to control the drone, it does not work properly and the drone just keeps flying following a given trajectory. This may be a problem of data conversion even if same data types are given in input to the Controller by using the same packages and functions that Dronati uses, but we still could not define the correct conversion if any. Another possible issue was that they implemented their package with a specific environment and, thus, adapting their code to it while we had to create it from pre-built assets from the Marketplace in a new empty project. This, in fact, could have led to setting mistakes also given by our inexperience with this tool.

As it was impossible for us to solve the problem related to AirSim, we implemented a new controller to replace Dronati's. This can be found in our repository scripts folder as `flight_prova.py` and it uses AirSim drone client to arm it, move it trough waypoints and make it land when done.

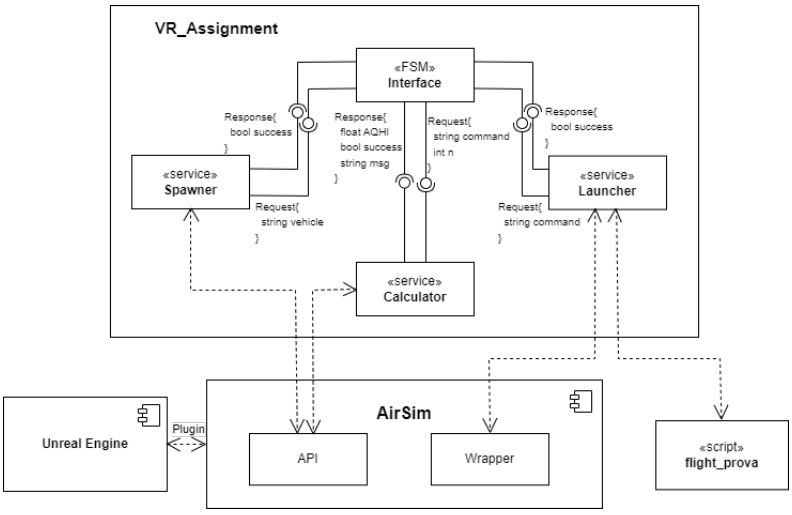

Figure 5: Final working architecture

We integrated it with very little changes in the already written code and, in fact, the architecture is not changed (Figure 5) as the `Launcher` simply runs this new script rather than Dronati package. Moreover, also each waypoint is passed to this controller with the same JSON file structure that we were expected to use for the other controller, proving how our architecture could have been robust and flexible.

As a result, we are able to successfully use the drone to sample the air and retrieve AQHI and health message for the citizens. This means we were able to satisfy the main objective of our project, but we cannot move the drone to one city location to another with the car. For this reason, in the following we are going to present what we obtained by running the simulation with our new controller.
In Figure 6 below it is possible to see how we located three pollution blocks above the PlayerStart (the highlighted frame in the bottom) where the drone is spawned whenever the simulation starts. We decided to draw the borders of each block just for visual purpose.

Figure 6: Screenshot of the city with the pollution blocks (three prisms) located above the PlayerStart.

In this configuration we run our software architecture in Figure 5 and we obtained an updated JSON file with many sections as the one in Listing 2 as many collision happened. Moreover, it prints on terminal the AQHI and corresponding health message whenever the user asks about it by entering 'X' as in the example below:

```
Getting Index
AQHI is calculated!
The AQHI is:  6.0
For at-risk individuals: adults and children with lung problems, and adults with heart problems, who
 experience symptoms, should consider reducing strenuous physical activity, particularly outdoors.
```

Figure 7: Example of AQHI and corresponding message displayed in the terminal after its computation

```
{
  "O3": 70.15534210205078,
  "SO2": 89.0661392211914,
  "NO2": 268.6379699707031,
  "PM10": 41.261329650878906,
  "PM25": 46.55827331542969,
  "waypoint": {
    "x": 0,
    "y": 0,
    "z": -5.024713516235352
  },
  "N": 1
}
}
```


Listing 2: Example of pollution sampling file updated with position and periodo of sampling.

6 Conclusions

6.1 Original implementation

As it was originally implemented, this project had some advantages with respect to how AirSim is used in literature:

1. Combining the FSM with AirSim API allows to use AirSim with multiple types of vehicles in the same simulation without the need to re-run it, which is actually not supported by AirSim originally. This would be a good achievement since multi-type vehicles is a mode frequently requested from developpers that want to use AirSim to control both cars and multirotors.

2. Another advantage is that since the algorithm relies on a FSM, it is possible to use this simulation ideally infinite times by moving around the city with the car and sampling the air with the drone continuosly. In this way, the simulation can also be considered more realistic.

On the other hand, there are also some limitation:

1. This approach works well whenever vehicles control is mutual, which means that it works specifically when car and multirotor don't have to be controlled at the same time. In other cases, e.g. Squadron of UAV and Ground Vehicles with SWARM technique, this approach cannot be used.

6.2 Final implementation

Despite the original implementation couldn't work properly, we manage to achieve our purpose to simulate the pollution sampling procedure for a Smart City by employing E-Drones. In the following we explain some future developments that can be done to the project.
One important improvement for this solution is related to the calculation of the AQHI by focusing on two aspects. The first is to define more precise values for each pollutants based on more specific parameters that an expert could give rather than generating them randomly. For example, each pollutant can be found in different molar concentration in different heights or near specific buildings and traffic conditions. Moreover, also information about time of day, temperature and humidity could be interesting to add for statistical purposes. The other aspect is related to sampling over time because the AQHI actually takes into consideration measurements in different time periods and city areas while we applied it to values sampled in only one period because we were not able to control both car and drone.
Another aspect that could be investigated is focusing on other types of pollution to sample with drones and ground vehicles as the waste, energy production, water and soil ones. Regarding more specifically the first, using a camera on the drone to take photos at each sample could also give some kind of information about pollution in defined area of the city and what may cause it so that a quick solution can be applied.
Just for sake of similarity with real world, another improvement could be to populate this simulation with people and vehicles moving throughout the city.

6.3 Application for educational purpose

Starting from this final implementation, an innovative kids videogame for educational purpose could be developed to make them become aware about civic educations in pollution context.
After the city is populated with people and vehicles to make it more realistic, many scenarios could be created while the gamer controls with an RC or the keyboard the vehicle around the city. For example, a scenario could display a recycling area in which the garbage is scattered all around the place and, thus, the gamer enters a level in which he/she is asked about which action is better to perform among some prompted options. Based on which action the gamer chooses and if this action is civically good or not, he/she gets a points award or penalty respectively. In any case, the gamer should be able to calculate the AQHI and gets the corresponding health message whenever he/she wants.
Finally, many animations could be created for as many actions as a level suggests to make the game more appealing and enjoyable for kids.

References

[1] Biosost website. https://www.biosost.com/.

[2] Oscar Alvear, Nicola Roberto Zema, Enrico Natalizio, Carlos T Calafate, et al. Using uav-based systems to monitor air pollution in areas with poor accessibility. *Journal of advanced Transportation*, 2017, 2017.

[3] Elena SF Berman, Matthew Fladeland, Jimmy Liem, Richard Kolyer, and Manish Gupta. Greenhouse gas analyzer for measurements of carbon dioxide, methane, and water vapor aboard an unmanned aerial vehicle. *Sensors and Actuators B: Chemical*, 169:128–135, 2012.

[4] Wenwei Che, Yumiao Zhang, Changqing Lin, Yik Him Fung, Jimmy C.H. Fung, and Alexis K.H. Lau. Impacts of pollution heterogeneity on population exposure in dense urban areas using ultra-fine resolution air quality data. *Journal of Environmental Science*, 125:513–523, 2023.

[5] EpicGames. Unreal engine.

[6] Alexander Malaver, Nunzio Motta, Peter Corke, and Felipe Gonzalez. Development and integration of a solar powered unmanned aerial vehicle and a wireless sensor network to monitor greenhouse gases. *Sensors*, 15(2):4072–4096, 2015.

[7] Microsoft. Aisim.

[8] World Health Organization. Air quality and health.

[9] Godall Rohi, O'tega Ejofodomi, and Godswill Ofualagba. Autonomous monitoring, analysis, and countering of air pollution using environmental drones. *Heliyon*, 6(1):e03252, 2020.

[10] ROS. Ros:home.

[11] ROS. smach - roswiki.

[12] Jose Ruiz-Jimenez, Nicola Zanca, Hangzhen Lan, Matti Jussila, Kari Hartonen, and Marja-Liisa Riekkola. Aerial drone as a carrier for miniaturized air sampling systems. *Journal of Chromatography A*, 1597:202–208, 2019.

[13] Shilpa Sonawani and Kailas Patil. Air quality measurement, prediction and warning using transfer learning based iot system for ambient assisted living. *International Journal of Pervasive Computing and Communications*, 2023.

[14] Waqi. A beginner's guide to air quality instant-cast and now-cast.

[15] Waqi. World's air pollution:real-time air quality index.

[16] Wikipedia. Air quality index.

[17] Wikipedia. Nowcast (air quality index).

[18] Xilin Yang, Luis Mejias Alvarez, and Matt Garratt. Multi-sensor data fusion for uav navigation during landing operations. In *Proceedings of the 2011 Australasian Conference on Robotics and Automation (ACRA 2011)*, pages 1–10. Australian Robotics & Automation Association, 2011.