UNIVERSITÀ
DEGLI STUDI
DI PADOVA


DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

# ACQUISITION AND PROCESSING SOFTWARE FOR AIR POLLUTION MEASUREMENT ON BOARD DRONES OF A.R.I.A. PROJECT

**Thesis supervisor: Prof. Carlo Bettanini Fecia di Cossato**

**Candidate: Giacomo Favaron**

**ACADEMIC YEAR: 2020-2021**

**Graduation Date: 11/15/2021**

**Abstract**

In the recent years, awareness of the issue of Environmental Pollution has increased, and research shows that not enough has been done, until now, to reduce pollution. In this domain, the monitoring of air quality is fundamental to provide data which can be used to most effectively guide our efforts to reduce air pollution. At this time the monitoring of air quality is usually performed via stationary ground-mounted air pollution stations. However, research [1][2] has shown that air pollution can vary greatly at different heights, for this reason the *Air Pollutants Monitoring Using UAVs* (ARIA) project is aiming to develop a system to measure vertical gradients of air pollutants using vertical swarms of drones. The ARIA project solution is a low-cost monitoring system based on *Commercial off-the-shelf* (COTS) sensors and on multiple cheap drone platforms. The system is equipped with $PM_2.5$ and $PM_10$ sensors to monitor the particulate concentration and several other gas sensors (such as $NO$, $NO_2$, $CO$, etc.) and the use of *Unmanned Aerial Vehicles* (UAVs) allows to build a 3D map of pollutants in a specific area. This could prove very useful around buildings in urban areas and possibile polluting plants in industrial areas. In this thesis are presented the system platform, the software implementation and a field test.

# Contents

# List of Figures

# List of abbreviations

**ADC** *Analog to Digital Converter*.

**ARIA** *Air Pollutants Monitoring Using UAVs*.

**COTS** *Commercial off-the-shelf*.

**EPA** *Environmnetal Protection Agengy*.

**GCS** *Ground Control Station*.

**GSO** *Ground Station Operator*.

**UART** *Universal Asynchronous Receiver-Transmitter*.

**UAVs** *Unmanned Aerial Vehicles*.

**WSN** *Wireless Sensor Network*.

# Chapter 1

# Introduction

Air pollution is caused by different typologies of gas pollutants that are present in the first meters (¡ 150 m) of the atmosphere and cause therefore damages to humans and environment. As air pollution is becoming the largest environmental health risk, the monitoring of air quality has drawn much attention in both laboratory studies and specific field tests and data collection campaigns. Government agencies and local administrations have, generally, provided and used monitoring stations on dedicated sites in cities and urban areas. Usually the studies have been conducted using fixed stations that are very reliable but produce only coarse-grained 2D monitoring, with several kilometers between two monitoring stations; or the stations monitor the same local area for long periods. Other approaches show that applications using simple system of sensors have been developed to monitor the fine-grained air quality using densely deployed sensors [3][4]. In any case, the fixed sensor station may achieve high precision, but have high cost and require maintenance and suffer especially for lack of mobility. Furthermore, these approaches don't account for the vertical gradients of air pollution levels. As shown in research [1][2] the concentrations of air pollutants can vary greatly at different heights and this is a sensitive factor in circumstances such as buildings in urban areas and possible polluting plants in industrial areas. The usage of Unmanned Aerial Vehicles (UAVs) has been particularly rich in the latest years due to their flexibility, mobility and affordable cost. Current monitoring systems are not able to satisfy every need of modern cities and industrial areas and UAVs are valuable supporting elements in this scenario. In terms of urban conditions, which is the main subject of the present study, UAVs can be used to measure environmental parameters such as illumination, wind speed, temperature, humidity, air quality and much more. In any case, for a complete analysis, both ground sensing and aerial sensing are necessary to provide 3D mapping and gas profiling. In our ARIA project, we equipped with the same set of sensors the devices that execute sensing on the ground, and the systems that execute aerial sensing on board the UAVs, which we are deploying in vertical swarms, to measure pollution levels at different heights. The fixed ground sensing suite is able to collect data in a continuous way, but the air quality of the higher levels of air off the ground cannot be detected, so the contemporary use of drones is mandatory. Aerial sensing, on the contrary, is able to sense the air quality off the ground, but it cannot be executed for very long periods due to the high consumption of battery power and human time. By merging the potentialities of these two systems of sensing suites, a better set of data can be collected. A trade off on the possible sensors and UAVs has been performed and quadcopters are the preferred platform for monitoring because of their simplicity, low cost and hovering capabilities. On the contrary a possible bias of data is due to the the influence of air jets created by the rotor rotation or by the electromagnetic field generated

by the antennas present on board. The problem of choosing the best location of the sensors is examined in [5] based on the physical structure of the drones. Our approach is to use an extension on which we fix the sensors in order to suck the air away of the main air jets.

## 1.1 Related works and state of the art

### 1.1.1 Air Pollution

According to [6] "Air pollution can be defined as the presence of toxic chemicals or compounds (including those of biological origin) in the air, at levels that pose a health risk. In an even broader sense, air pollution means the presence of chemicals or compounds in the air which are usually not present and which lower the quality of the air or cause detrimental changes to the quality of life (such as the damaging of the ozone layer or causing global warming)". Air pollution is extremely complex to evaluate and there are many polluting substances in the atmosphere. The *Environmnetal Protection Agengy* (EPA) (of United States) takes these 6 (the "criteria air pollutants") in consideration in its studies:

Table 1.1: Criteria air pollutants and their health effects [7]

| Chemical symbol | Substance | Characteristics | Effect |
|---|---|---|---|
| CO | Carobon Monoxide | Colorless, odorless gas | Reducing oxygen delivery to the body's orogans and tissues |
| $NO_2$ | Nitrogen Dioxide | Highly reactive gas | Risk of emphysema, asthma and bronchitis diseases |
| $O_3$ | Ozone | Pale blue gas | Chest pain, coughing, throat irritation |
| $SO_2$ | Sulfur Dioxide | Colorless, irritating smell gas | Risks of bronchoconstriction and increase of asthma symptoms |
| $PM_{2.5}$ and $PM_{10}$ | Particulate Matter | Inhalable particles | Premature death and respiratory symptoms |
| Pb | Lead | Metal particles | Accumulate in bones and affecting the nervous system |

### 1.1.2 Air quality monitoring

Air quality monitoring is an essential part in order to know what measures to put in place [8] to protect our health and the environment, which are strongly connected. In the Veneto region, Italy, the area of this study, the ARPAV[9] has put in place a conventional monitoring network to track major air pollutants and enforce restrictory measures on polluting factors (such as transportation) if necessary. Figure 1.1 shows the map of ARPAV's current 2D monitoring network, which gives an example of the very low spacial resolution of conventional air quality monitoring systems.

### 1.1.3 Low-cost sensors

The EPA also provides the Air Sensor Guidebook[10] which gives extensive information on air quality and low-cost sensors. Due to their prohibitive cost and complexity, conventional air pollution monitoring systems have low spacial and temporal resolution. Low-cost sensors, instead, can be deployed more diffusely with high spacial and temporal resolution, while trading most of their accuracy. They are, in fact, heavily influenced by many

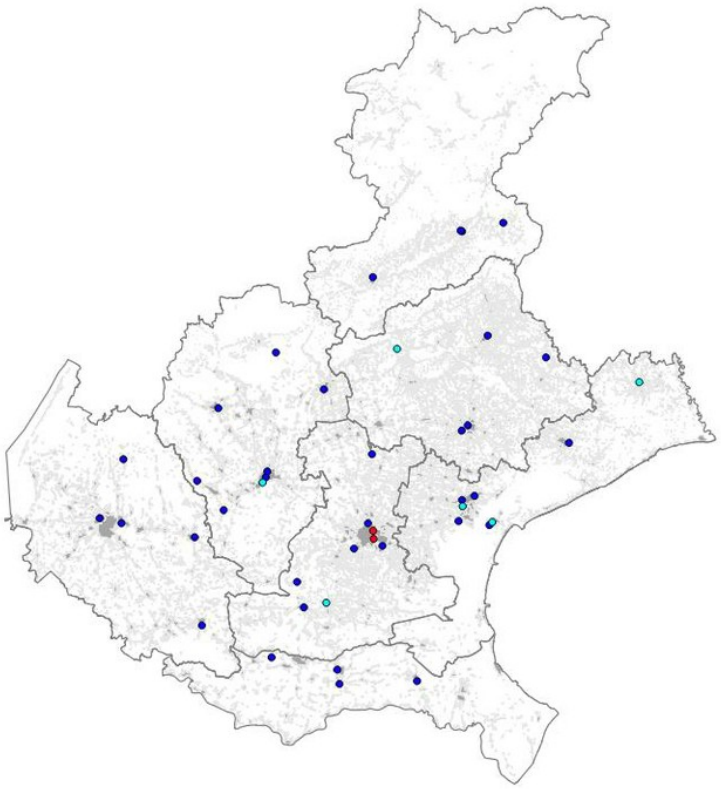


Figure 1.1: ARPAV monitoring netowrk in the Veneto region, Italy[9]

factors, especially temperature, humidity, wind and presence of other gases in the air. Due to their lightweight, only low-cost sensors can be mounted on drones: the aforementioned influence factors could introduce even more inaccuracy, especially wind generated from the rotors. [11] among other things presents an evaluation of air quality sensors classifying their performances by the most important parameters.

### 1.1.4 Drone systems

Coordinating data collection and movements is quite hard, considering the low computational power of UAVs, the inaccuracy of GPS and the general wireless communication issues. [11] is a survey on the communication issues releted to drone networks. The energy constraint is really impactful on long missions and poses a major limitation to the spreading of drone technology. Drones have excellent mobility and data gathering prowess, but cannot always rely on coming back to the base to deliver their information. Implementing reasonable communication protocols and algorithms is necessary to improve efficiency.

### 1.1.5 Drone monitoring and sensing

Thanks to their mobility and flying movement, drone monitoring and sensing capabiliteis are very valuable. In urban settings UAVs can monitor noise, traffic, light, wind, temperature, humidity, air quality and many other parameters. As shown earlier, conventional air quality monitoring systems have very low spatial and temporal resolution. UAVs based systems could measure specific areas with great convenience and felxibility and hybrid ground and air based solutions could routinely track the air pollution levels of parts of a city. A single UAV,

however, is very limited in its performance due to its coverage, energy autonomy and small selection of sensors. Swarms, instead, can provide full coverage of an area, while coordinating the best routes to visit each sensing node. Equipping different drones with different sensors is far easier and more flexible than having one doing everything on its own. [7], [12], [13], [14] propose different solutions for a *Wireless Sensor Network* (WSN) using UAVs. [13] in particular examines an application in smart cities, where a hybrid ground and air based system tracks an urban area.

## 1.2 Dissertation structure

This dissertation describes the ARIA project solution for the monitoring of air pollution. It is divided into 6 chapters:

- Chapter 1 describes the introduction, an overview on the topic of air pollution, the motivation to approach the problem, the motivation of the proposed solution, related works, the state of the art and the dissertation objective.
- Chapter 2 describes the system architecture, that is the UAVs that are being used, their design, specifications and functionality.
- Chapter 3 describes the sensor payload, the motivation of the adopted sensors and their use.
- Chapter 4 describes the software implementation for the data collection of the sensors and the communication of the UAVs.
- Chapter 5 shows the results of a test flight using the proposed solution.
- Chapter 6 presents what conclusions can be taken after all the developed work, and what improvements can be done in the future.

## 1.3 Dissertation objective

The objective of this dissertation is to describe the solution proposed by the ARIA project for air pollution monitoring, in particular the software impelementation, to show preliminary results and discuss their revelance in future applications.

# Chapter 2

# ARIA project

## 2.1 Overview

ARIA project was created by a group of students from the Department of Industrial Engineering, University of Padova, under the suggestion and guidance of personnel staff of the Center for Space Studies and Activities (CISAS) of the same University. The core motivation that brought together these students was the desire of researching new fields of application for drone technology. ARIA project's scenario is investigation about drones usage within air quality monitoring. Environmental pollution is becoming every day more threatening for our health and we wanted to develop a tool to monitor it in 3D. the project is funded by the Department of Industrial Engineering of our University.

## 2.2 Proposal presentation

ARIA project's ultimate goal is gathering data about the values of major pollutants in urban areas at various heights. Our tool will be a vertical drone swarm equipped with low-cost sensors and deployable by utilizing GPS coordinates. The information will be stored in a database, so that they can be post-processed and published whenever appropriate. The most interesting activity could be creating a contemporary vertical profile of pollutants concentrations for various time-periods. We've chosen to work with UAVs because after reading literature about environmental monitoring, we found omissions in Wireless Sensor Networks. The use of this new technology was already recommended thanks to its speed, mobility and capability to fly at different heights. The main constraint was the unavailability of low-cost and low bulk drones. To differentiate ourselves from previous studies (since our knowledge would not be able to compete with them anyway) we wanted to explore vertical swarms. The study of pollution at different heights is not a popular topic and we wanted to investigate it. The task for the multicopter will have a standard implementation: usually it will be a simple request to move towards some GPS coordinates, hover there while collecting samples and then return to the base whenever the time is up or the battery is too low, or move through different GPS waypoints, hovering at each one to allow the sensors to adjust and collect the data. The gas sensors have a response time of around 30s which needs to be taken into consideration. The flight will be planned before departure from a terminal. The vertical deployment doesn't need to be extremely precise, since sensor's accuracy is not sufficient for slight misplacements to matter, and . Since our approach will be careful and gradual, each entity belonging to the

swarm will not communicate with the others. Considering this fact, we know the term "swarm" is being used inappropriately. Wireless communication and real swarm implementation will follow as the project unfolds.

# Chapter 3

# ARIA: System architecture

The selected system is a compromise between payload capacity, in-flight stability and manoeuvrability and low-cost. The system is composed by:

- Tarot 650 Sport drone, for the platform
- Hex Cube Black Flight Controller
- HERE2 GPS system
- Raspberry Pi3b+ for the controller devoted to sensor measurement
- A suite of sensors for air quality monitoring (in particular NO, NO2, CO and VOCs) based on Alphasesnse AFE board
- Nova SDS011 PM board to measure PM2.5 and PM10
- Taranis 9D+ Radio and an 8XR receiver

The ARIA Project is going to use simultaneously two drones and a fixed ground station, equipped with the same instrumentation in order two build a 3D map of the investigated area. We are currently experimenting with the use of an Arduino controller on the second drone to compare the data collected by a different platform and allow the presence of additional sensor (such as temperature) thanks to Arduino's support of an higher number of input channels. This dissertation will focus on the Rapberry based system. There has been just some laboratory testing on the effect of the blade disturbances on the measurement [5] but a comprehensive study on sensor location on such platforms has not been done yet; therefore the ARIA project has designed a vertical boom were all the gas sensors are located. On the other hand, in the area above the UAV, there is a relatively constant air flow which drops significantly after a distance of approximately 40.0-50.0 cm for devices with characteristics similar to those used in the proposed solution. The airflow behaviour is similar aside from the UAV in the area with a radius from the center $r > 50.0$ cm. To avoid the swirl area, it is recommended to place horizontal and vertical probes of the appropriate length. The use of horizontal probes often makes it difficult to achieve the conditions necessary for isokinetic sampling. In addition, it is necessary to use additional structures for their equalization. To overcome this problem, it was decided to use the vertical probe; the use of a boom in a vertical position does not affect the flight capabilities of the UAV since the entire system has a low center of mass, situated in the proximity of the battery. Figure 3.1 contains an overview of the various

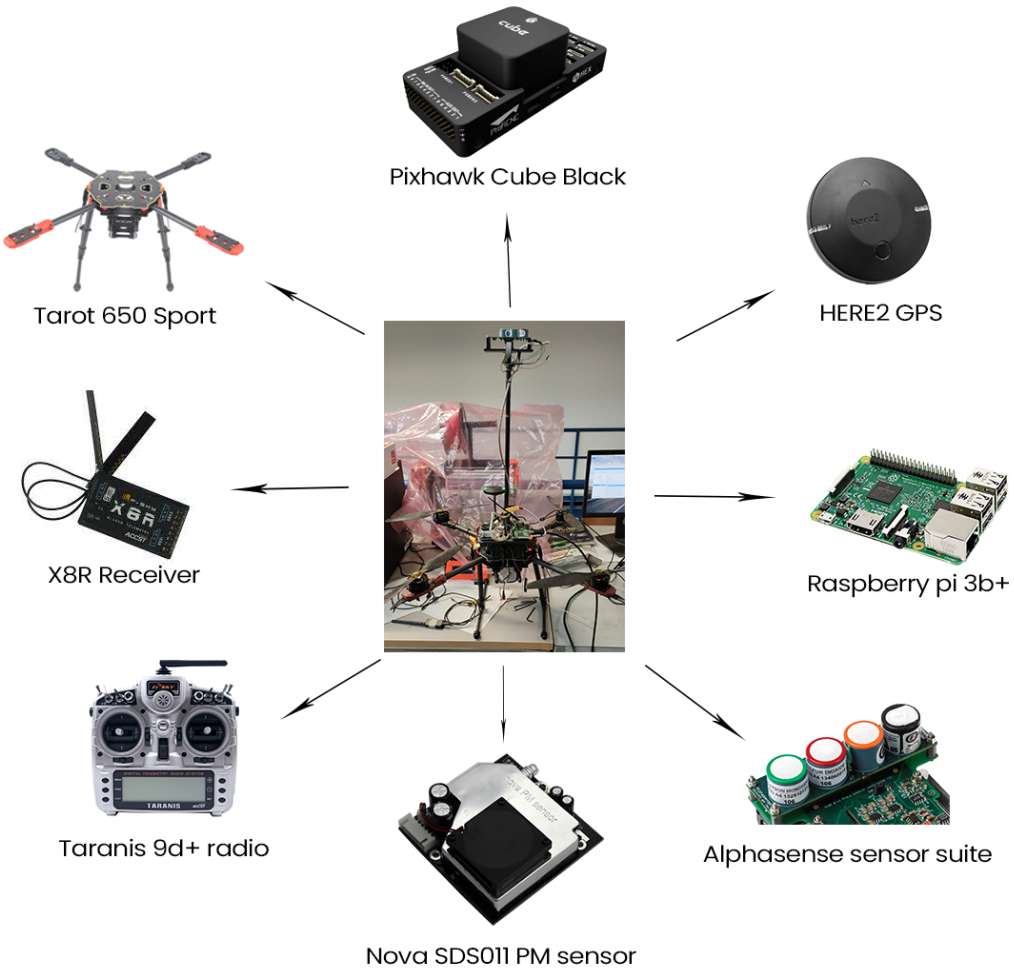
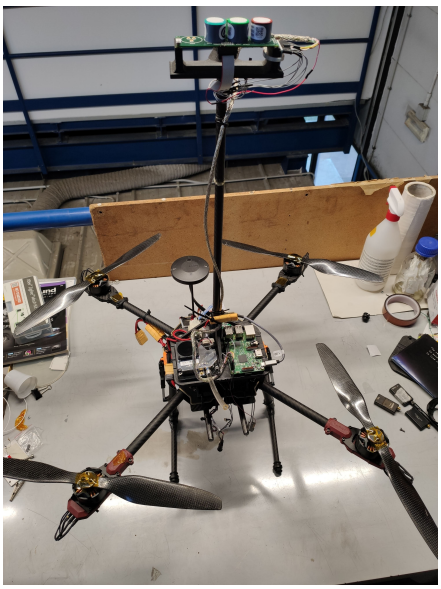
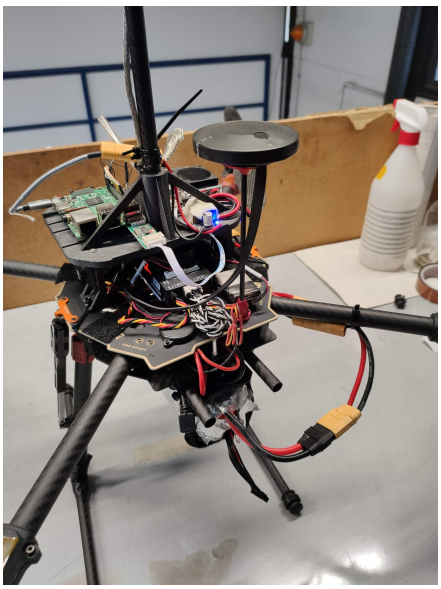



Figure 3.1: ARIA drone system components

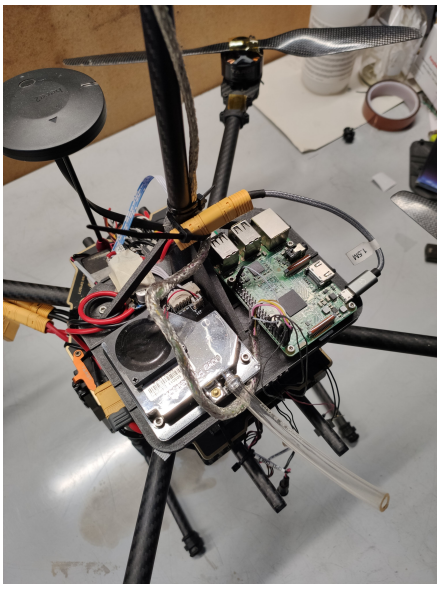
system components; while Figure 3.2 shows one of the drones and some details of the system. The UAV pilot or the UAV *Ground Station Operator* (GSO) can communicate with the UAV wirelessly, using a radio controller (RC) transmitter or a computer, respectively. Power is provided by a 6s Lipo battery able to give 10000mAh. Communication between the flight controller and ground station is via 433MHz telemetry link connected to a laptop; a 2.4GHz communication link is also ensured via the Taranis 9D+ Radio and an 8XR receiver onboard the UAV.



(a)



(b)



(c)

Figure 3.2: The Rasperry Pi ARIA drone

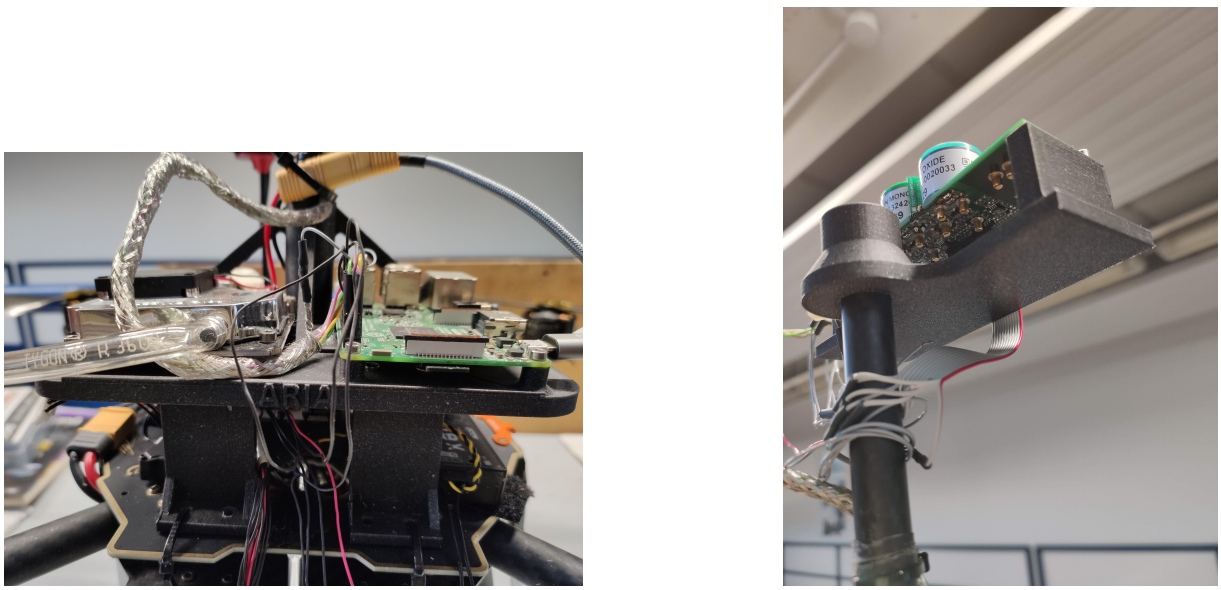
# Chapter 4

# Sensor Payload

The sensor payload is composed by 3 subsystems:

1. Raspberry Pi 3b+
2. Analogue Front End (AFE) by Alphasense equipped with 3 gas sensors (NO, NO2 and CO) and a PID (VOCs) sensor
3. Nova SDS011 particulate sensor board for PM2.5 and PM10

A student bachelor thesis performed a FEM study on the sensors and the Pi3b+ support platforms. The elements have been studied in order to present a high resistance to vibrations and to allow the sensors and the boards to be housed in the most safest way. In addition, the study also sought to design very light components: the boards were built with a 3D PLA printer in order to host in a compact and reliable way all the instrumentation (see Figure **??**).



(a) (b)

Figure 4.1: The 3D printed boards

## 4.1 Particulate sensors

The particulate sensor is a Nova SDS011 PM sensor able to measure PM2.5 and PM10 concentrations with a resolution of $0.3\mu g/m^3$; range of measurement is: $0-1mg/m^3$ and the frequency of output is 1 Hz. The sensor is a COTS sensor used for house and environmental monitoring with consumption around 1 W. Tests on the sensor were performed in the lab and outputs were calibrated. The sensor is equipped with a 10 cm inlet pipe (see Figure 2) in order to facilitate the inflow of the air in the sensor. Sensor is linked to the Raspberry via a USB interface board allowing fast connection and is located on the drone main platform.

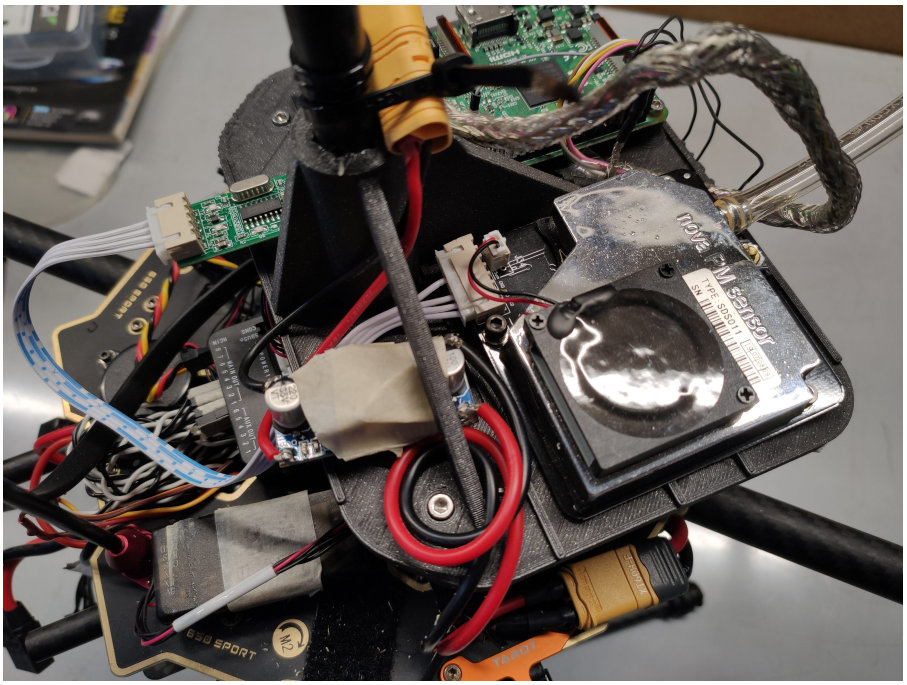


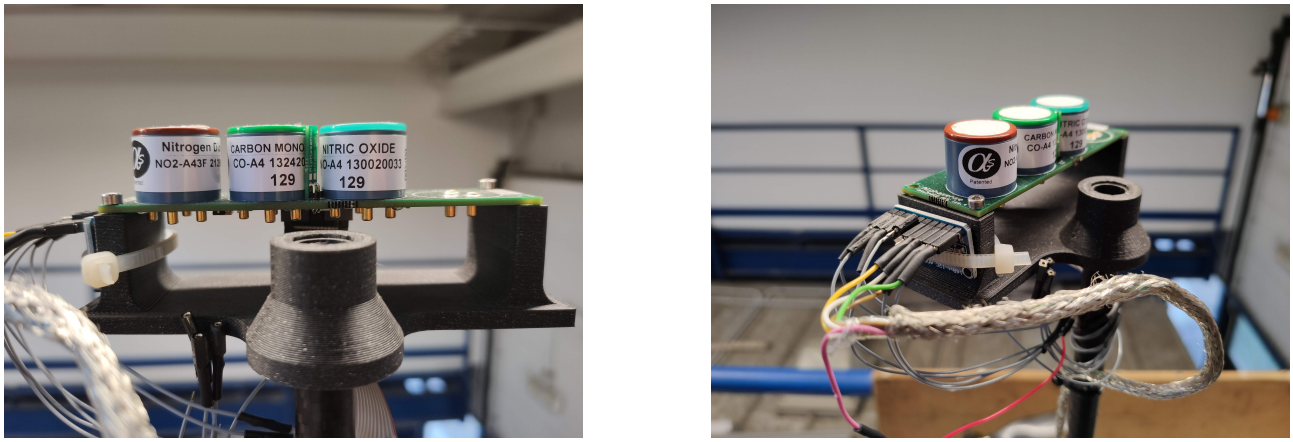

Figure 4.2: The SDS011 particulate sensor and it's USB connected to the Raspberry Pi (partially covered by the pole structure)

## 4.2 Gas sensors

Table 4.1: ARIA's gas sensors

| Substance | Sensor | Weight [g] | Range [ppb] | Data sheet |
|---|---|---|---|---|
| $CO$ | Alphasense CO-A4 | $< 6$ | 500 | [15] |
| $NO$ | Alphasense NO-A4 | $< 6$ | 20 | [16] |
| $NO_2$ | Alphasense NO-A43F | $< 6$ | 20 | [17] |
| $VOCs$ | Alphasense PID-AH2 | $< 8$ | 40 (isobutylene) | [18] |

They are connected to the Adafruit ADS1115 ADC and then to the raspberry. Due to The gas sensors have been positioned on a 50 cm boom in the center of the drone main platform and above the drone itself (see Figure 2) in order to reduce disturbances due to variable flux moved by the blades. Preliminary analysis show that the sensors are located in a non-altered flux allowing for reliable measurements. Due to possible disturbances from electromagnetic sources present on the drone (mainly telemetry antennas and electronics) a Faraday cage, not present in this study, but will be built around the gas sensors and tested in future flights.




(a) (b)

Figure 4.3: The Alphasense gas sensors on the AFE board on top of the pole and the ADS1115 ADC on the side. The NO sensor is present but not connected.

# Chapter 5

# Flight Control

There are two computers on board the ARIA system:

- Hex Cube Black flight controller, to control the UAV.
- Raspberry pi 3b+ as a companion computer, for additional communication with the *Ground Control Station* (GCS) and the sensor data acquisition.

## 5.1 Hex Cube Black

The Cube autopilot is a flexible autopilot intended primarily for manufacturers of commercial systems. It is based on the Pixhawk-project (opens new window) FMUv3 open hardware design [19]. It is the core of the UAV, all the main components needed for the flight (rotors, battery, receiver) are connected to its board and it handles all the processing needed to control the UAV.

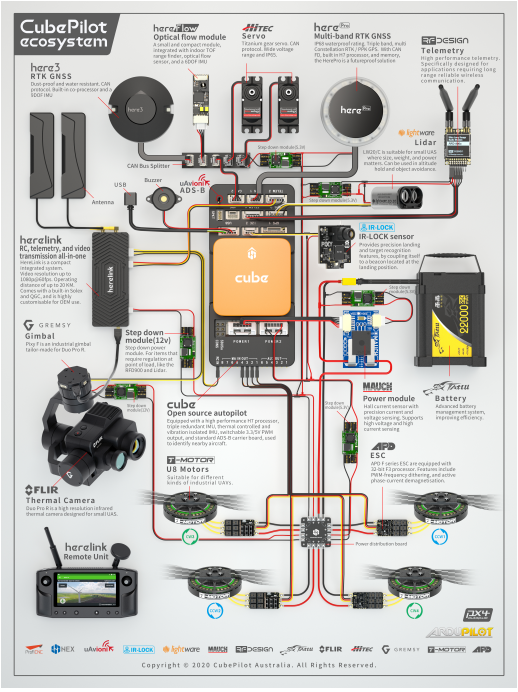



Figure 5.1: The Cubepilot Ecosystem[19]

The Hex Cube Black is an ArduPilot compatible autopilot.

## 5.2 ArduCopter

The firmware that the Pixhawk runs is the Arducopter, the copter specific version of ArduPilot, it is an open source firmware that supports the fully autonomous waypoint based flight and real time control, using MAVLink protocol to communicate with the Remote GCS. The GCS software used is Mission Planner.

## 5.3 Mission Planner

Mission Planner is a full-featured ground station application for the ArduPilot open source autopilot project [19]. It is used to configure the vehicle and plan the missions.


Figure 5.2: Screenshot of Mission Planner main screen, the gps is fixed in the laboratory location and the telemetry data is shown.

# Chapter 6

# Sensor Data Acquisition

In the ARIA project system the air pollution sensors are connected to the Raspberry pi companion computer, which reads the data and saves it along with the telemetry data received from the autopilot. My role when I joined the ARIA project was writing the necessary code to achieve this reading and saving. Figure 6.1 shows a diagram of the sensors architecture.

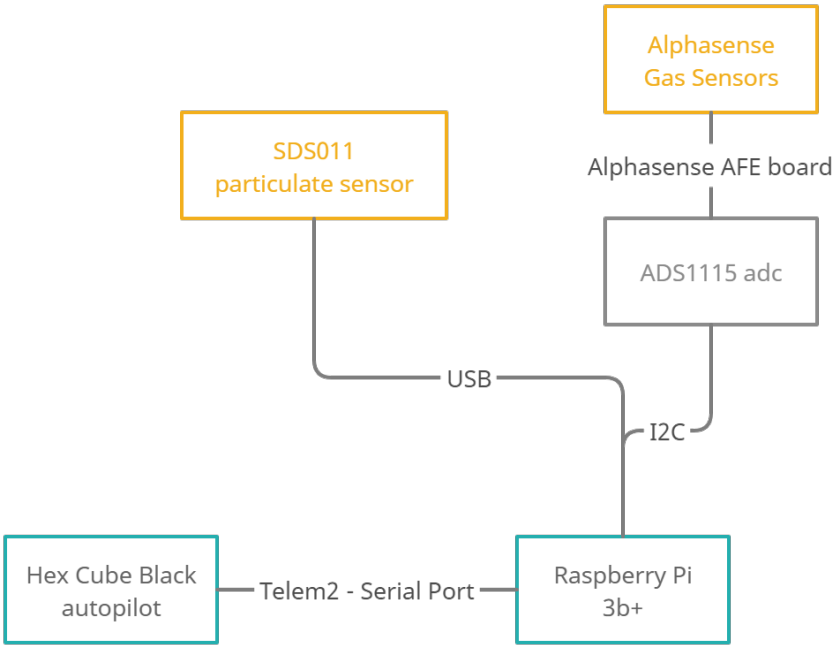


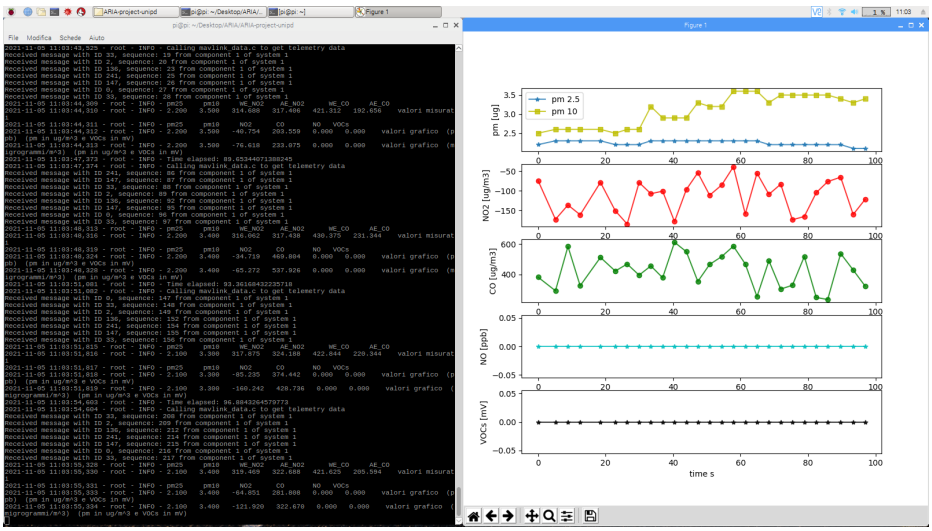

Figure 6.1: ARIA sensors architecture

The task can be divided into 2 main parts:

1. Reading and saving the data from the gas and particulate sensors.
2. Connecting the Raspberry Pi to the pixhawk autopilot and receive the telemetry data through the MAVLink protocol.

## 6.1 Sensor data reading and saving

Figure 6.1 also shows the connections between the various components of the system. The gas sensors are connected to the Raspberry Pi via the ADS115 *Analog to Digital Converter* (ADC) which is in turn connected to the i2c bus on the Raspberry. The SDS011 particulate sensor is instead connected via USB. The code to read from these sensorse was written in Python3 (from now on referred to as Python for simplicity) and uses the official Adafruit Python ADS1x15 library and a custom Python based client for the SDS011 sensor. The script (`aria.py`), after checking the connection with the components, continuously reads the data from the sensors. In the case of the gas sensors data received from the ADS1115, the raw data is also directly processed using specific formulas which were sent to us by Alphasense (Alphasense AAN 803-05, see Appendix A for specifications). These formulas take in input the raw electrode readings (in mV, two per gas sensor) and output the measured gas concentration (in ppb), and include a series of parameters which depend on the specific serial number of the sensor and on the external temperature. Currently the Raspberry Pi based UAV only reads data from the NO2 and CO sensors since the ADS1115 only supports 4 channels and 2 channels are required for each sensor (for the two electrodes, "AE" and "WE" in the formulas). A larger module is going to be installed in the near future. The data is then plotted in real time using the Python pyplot library and saved to a two different csv files, one with the raw sensor data, and the other for the processed sensor data. Every row contains also the date and time, the elapsed time since the acquisition script was started and the UNIX system time which is useful to check the synchronization with the data coming from the autopilot 6.2. Figure 6.2 shows examples of the terminal output, the live plot and the output csv files.




(a) Screenshot of the terminal output and live plot while the script is running.

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | UNIX time | time since | lat | lon | alt | relative_a | vx | vy | vz | hdg | |
| 2 | 1.64E+15 | 3661777 | 4.54E+08 | 1.19E+08 | 30910 | -74 | -3 | 9 | 0 | 3224 | |
| 3 | 1.64E+15 | 3666777 | 4.54E+08 | 1.19E+08 | 30950 | -36 | 4 | 5 | 0 | 3198 | |
| 4 | 1.64E+15 | 3670276 | 4.54E+08 | 1.19E+08 | 30910 | -70 | -12 | 4 | 0 | 3178 | |
| 5 | 1.64E+15 | 3673778 | 4.54E+08 | 1.19E+08 | 30950 | -33 | -13 | 0 | 0 | 3163 | |
| 6 | 1.64E+15 | 3677283 | 4.54E+08 | 1.19E+08 | 30940 | -47 | 0 | -2 | 0 | 3149 | |
| 7 | 1.64E+15 | 3680780 | 4.54E+08 | 1.19E+08 | 30990 | 9 | 10 | -3 | -1 | 3136 | |
| 8 | 1.64E+15 | 3684276 | 4.54E+08 | 1.19E+08 | 31080 | 97 | 10 | -1 | -2 | 3125 | |
| 9 | 1.64E+15 | 3687777 | 4.54E+08 | 1.19E+08 | 31070 | 84 | 2 | 0 | 0 | 3116 | |
| 10 | 1.64E+15 | 3691784 | 4.54E+08 | 1.19E+08 | 31090 | 105 | -7 | 2 | 0 | 3107 | |
| 11 | 1.64E+15 | 3695278 | 4.54E+08 | 1.19E+08 | 31110 | 124 | -2 | 0 | 0 | 3098 | |
| 12 | 1.64E+15 | 3698781 | 4.54E+08 | 1.19E+08 | 31100 | 111 | -4 | 4 | 0 | 3091 | |
| 13 | 1.64E+15 | 3702283 | 4.54E+08 | 1.19E+08 | 30970 | -10 | 0 | 3 | 2 | 3082 | |
| 14 | 1.64E+15 | 3705782 | 4.54E+08 | 1.19E+08 | 31030 | 43 | 3 | -4 | 0 | 3074 | |
| 15 | 1.64E+15 | 3709288 | 4.54E+08 | 1.19E+08 | 31060 | 71 | -1 | -3 | 0 | 3066 | |
| 16 | 1.64E+15 | 3713290 | 4.54E+08 | 1.19E+08 | 31050 | 67 | -3 | 3 | 0 | 3060 | |
| 17 | 1.64E+15 | 3716781 | 4.54E+08 | 1.19E+08 | 31120 | 134 | -3 | 3 | -1 | 3057 | |
| 18 | 1.64E+15 | 3720280 | 4.54E+08 | 1.19E+08 | 31100 | 119 | -8 | 4 | 0 | 3051 | |
| 19 | 1.64E+15 | 3723776 | 4.54E+08 | 1.19E+08 | 31050 | 61 | 0 | -4 | 1 | 3044 | |
| 20 | 1.64E+15 | 3727276 | 4.54E+08 | 1.19E+08 | 30990 | 7 | 19 | -13 | 2 | 3036 | |
| 21 | 1.64E+15 | 3730776 | 4.54E+08 | 1.19E+08 | 31030 | 44 | 3 | -1 | 0 | 3036 | |
| 22 | 1.64E+15 | 3734280 | 4.54E+08 | 1.19E+08 | 31070 | 80 | 4 | -1 | -1 | 3033 | |
| 23 | 1.64E+15 | 3737778 | 4.54E+08 | 1.19E+08 | 31040 | 53 | 6 | 0 | -1 | 3030 | |
| 24 | 1.64E+15 | 3741280 | 4.54E+08 | 1.19E+08 | 30990 | 1 | -1 | -1 | 0 | 3027 | |
| 25 | 1.64E+15 | 3744784 | 4.54E+08 | 1.19E+08 | 30940 | -43 | -8 | 2 | 1 | 3028 | |
| 26 | 1.64E+15 | 3748283 | 4.54E+08 | 1.19E+08 | 31000 | 12 | -7 | 3 | 0 | 3027 | |
| 27 | 1.64E+15 | 3751778 | 4.54E+08 | 1.19E+08 | 30970 | -11 | -7 | 0 | 1 | 3027 | |
| 28 | 1.64E+15 | 3755280 | 4.54E+08 | 1.19E+08 | 30960 | -24 | -6 | 0 | 1 | 3025 | |
| 29 | 1.64E+15 | 3758783 | 4.54E+08 | 1.19E+08 | 31030 | 48 | 0 | 0 | 0 | 3025 | |
| 30 | 1.64E+15 | 3762284 | 4.54E+08 | 1.19E+08 | 30930 | -59 | 3 | -4 | 1 | 3025 | |
| 31 | 1.64E+15 | 3765782 | 4.54E+08 | 1.19E+08 | 30990 | 9 | 11 | -5 | 0 | 3025 | |

mavlink_output2021_10_25_15_00_

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | time_elap | pm25 [ug], | pm10 [ug], | NO2 [ug/r | CO [ug/m./ | | / | data | ora | unix time | |
| 2 | 7.15E-06 | 3.6 | 10.1 | 37.10952 | 52.67795 | 0 | 0 | 25/10/202 | 15:00:39 | 1.64E+09 | |
| 3 | 4.926795 | 3.6 | 9.7 | 62.36124 | -150.153 | 0 | 0 | 25/10/202 | 15:00:44 | 1.64E+09 | |
| 4 | 8.444927 | 3.7 | 9.7 | -39.0768 | 151.0054 | 0 | 0 | 25/10/202 | 15:00:48 | 1.64E+09 | |
| 5 | 11.82686 | 3.7 | 9.8 | -44.844 | -2.12984 | 0 | 0 | 25/10/202 | 15:00:51 | 1.64E+09 | |
| 6 | 15.43667 | 3.8 | 10.5 | 22.04472 | 229.4544 | 0 | 0 | 25/10/202 | 15:00:55 | 1.64E+09 | |
| 7 | 18.98546 | 3.7 | 10.2 | -9.0281 | 229.0994 | 0 | 0 | 25/10/202 | 15:00:58 | 1.64E+09 | |
| 8 | 22.44097 | 3.7 | 10.3 | 43.98165 | 256.1483 | 0 | 0 | 25/10/202 | 15:01:02 | 1.64E+09 | |
| 9 | 25.90318 | 3.6 | 10 | -13.8251 | 266.7975 | 0 | 0 | 25/10/202 | 15:01:05 | 1.64E+09 | |
| 10 | 29.42781 | 3.7 | 9.9 | -19.4576 | 272.974 | 0 | 0 | 25/10/202 | 15:01:09 | 1.64E+09 | |
| 11 | 33.38633 | 3.8 | 10.6 | 28.16227 | 309.4652 | 0 | 0 | 25/10/202 | 15:01:13 | 1.64E+09 | |
| 12 | 36.94118 | 3.7 | 10.6 | 41.69094 | 145.3258 | 0 | 0 | 25/10/202 | 15:01:16 | 1.64E+09 | |
| 13 | 40.47804 | 3.8 | 10.6 | -2.668 | 68.29675 | 0 | 0 | 25/10/202 | 15:01:20 | 1.64E+09 | |
| 14 | 43.96737 | 3.8 | 10.6 | 55.48911 | 271.6251 | 0 | 0 | 25/10/202 | 15:01:23 | 1.64E+09 | |
| 15 | 47.35363 | 3.7 | 10.7 | 72.22477 | 266.1585 | 0 | 0 | 25/10/202 | 15:01:26 | 1.64E+09 | |
| 16 | 51.38905 | 3.9 | 10.9 | -49.1829 | 170.1029 | 0 | 0 | 25/10/202 | 15:01:31 | 1.64E+09 | |
| 17 | 54.90268 | 3.8 | 10.7 | 26.68005 | 162.6485 | 0 | 0 | 25/10/202 | 15:01:34 | 1.64E+09 | |
| 18 | 58.41087 | 3.9 | 11.5 | 12.63933 | 237.9737 | 0 | 0 | 25/10/202 | 15:01:38 | 1.64E+09 | |
| 19 | 61.95271 | 4 | 11.5 | -41.1519 | 247.629 | 0 | 0 | 25/10/202 | 15:01:41 | 1.64E+09 | |
| 20 | 65.44641 | 3.9 | 11.3 | 68.20929 | 46.64342 | 0 | 0 | 25/10/202 | 15:01:45 | 1.64E+09 | |
| 21 | 68.94037 | 3.9 | 11.1 | 21.15539 | 219.3731 | 0 | 0 | 25/10/202 | 15:01:48 | 1.64E+09 | |
| 22 | 72.46653 | 3.9 | 10.9 | 9.620986 | 186.9996 | 0 | 0 | 25/10/202 | 15:01:52 | 1.64E+09 | |
| 23 | 75.9751 | 3.7 | 7.8 | 58.9117 | 293.8464 | 0 | 0 | 25/10/202 | 15:01:55 | 1.64E+09 | |
| 24 | 79.45667 | 3.8 | 7.8 | -29.375 | 219.5861 | 0 | 0 | 25/10/202 | 15:01:59 | 1.64E+09 | |
| 25 | 82.98513 | 3.8 | 8.1 | -32.7167 | 257.7102 | 0 | 0 | 25/10/202 | 15:02:02 | 1.64E+09 | |
| 26 | 86.38967 | 3.8 | 7.8 | 99.79415 | 218.1662 | 0 | 0 | 25/10/202 | 15:02:06 | 1.64E+09 | |
| 27 | 89.89922 | 3.8 | 8 | -13.2322 | 116.3601 | 0 | 0 | 25/10/202 | 15:02:09 | 1.64E+09 | |
| 28 | 93.40701 | 3.9 | 8.1 | 21.90998 | 268.8563 | 0 | 0 | 25/10/202 | 15:02:13 | 1.64E+09 | |
| 29 | 96.87257 | 3.9 | 8.2 | 45.54472 | 96.41059 | 0 | 0 | 25/10/202 | 15:02:16 | 1.64E+09 | |
| 30 | 100.3626 | 4 | 8.4 | -49.2099 | 211.7767 | 0 | 0 | 25/10/202 | 15:02:20 | 1.64E+09 | |
| 31 | 103.9395 | 4 | 10.7 | 4.985665 | 215.6814 | 0 | 0 | 25/10/202 | 15:02:23 | 1.64E+09 | |

misuredroneProc2021_10_25_15_00

(b) Example of the output csv files, in particular the one with the telemetry data on the left, and the one with the sensors data on the right

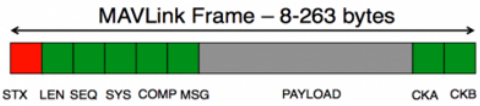
Figure 6.2: Examples of terminal output, live plot and output csv files from the main script aria.py

## 6.2 Telemetry data

To facilitate the processing of the data acquired during flight, we wanted to receive the telemetry data from the autopilot and temporally link it to the sensors data during flight. This would remove the need of a subsequent synchronization. To achieve this we connected the autopilot to the Raspberry Pi, which are both on board of the drone. As shown in Figure 6.1 the Telem2 port on Hex Cube Black autopilot is connected to a serial port on the Raspberry Pi. Telem2 is the port on the Pixhawk for MAVLink communication.

### 6.2.1 MAVLink protocol

MAVLink is a very lightweight messaging protocol for communicating with drones and between onboard drone components. It is designed as a header-only message marshaling library.[20].




| Byte Index | Content | Value | Explanation |
|---|---|---|---|
| 0 | Packet start sign | v1.0: 0xFE (v0.9: 0x55) | Indicates the start of a new packet. |
| 1 | Payload length | 0 - 255 | Indicates length of the following payload. |
| 2 | Packet sequence | 0 - 255 | Each component counts up his send sequence. Allows to detect packet loss |
| 3 | System ID | 1 - 255 | ID of the SENDING system. Allows to differentiate different MAVs on the same network. |
| 4 | Component ID | 0 - 255 | ID of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot. |
| 5 | Message ID | 0 - 255 | ID of the message - the id defines what the payload "means" and how it should be correctly decoded. |
| 6 to (n+6) | Data | (0 - 255) bytes | Data of the message, depends on the message id. |
| (n+7) to (n+8) | Checksum (low byte, high byte) | ITU X.25/SAE AS-4 hash, **excluding packet start sign, so bytes 1..(n+6)** Note: The checksum also includes MAVLINK_CRC_EXTRA (Number computed from message fields. Protects the packet from decoding a different version of the same packet but with different variables). | |

Figure 6.3: MAVLink packet structure[19]

Pixhawk and ArduPilot use MAVLink for communication. It is used between the Mission Planner GCS software and the pixhawk, to control the UAV; and in the serial connection between the pixhawk and the Raspberry Pi companion computer, to send telemetry data. MAVLink can transport messages and commands which are defined in a *dialect*. We used the common dialect defined in the official documentation[21], and the messages we needed were 'SYSTEM_TIME' (ID #2) and 'GLOBAL_POSITION_INT' (ID #33). They, in fact, carry the UNIX time of the autopilot and telemetry data such as position, altitude and velocity. Figure 6.4 shows the exact content of the two messages.

### SYSTEM_TIME ( #2 )

[Message] The system time is the time of the master clock, typically the computer clock of the main onboard computer.

| Field Name | Type | Units | Description |
|---|---|---|---|
| time_unix_usec | uint64_t | us | Timestamp (UNIX epoch time). |
| time_boot_ms | uint32_t | ms | Timestamp (time since system boot). |

### GLOBAL_POSITION_INT ( #33 )

[Message] The filtered global position (e.g. fused GPS and accelerometers). The position is in GPS-frame (right-handed, Z-up). It is designed as scaled integer message since the resolution of float is not sufficient.

| Field Name | Type | Units | Description |
|---|---|---|---|
| time_boot_ms | uint32_t | ms | Timestamp (time since system boot). |
| lat | int32_t | degE7 | Latitude, expressed |
| lon | int32_t | degE7 | Longitude, expressed |
| alt | int32_t | mm | Altitude (MSL). Note that virtually all GPS modules provide both WGS84 and MSL. |
| relative_alt | int32_t | mm | Altitude above ground |
| vx | int16_t | cm/s | Ground X Speed (Latitude, positive north) |
| vy | int16_t | cm/s | Ground Y Speed (Longitude, positive east) |
| vz | int16_t | cm/s | Ground Z Speed (Altitude, positive down) |
| hdg | uint16_t | cdeg | Vehicle heading (yaw angle), 0.0..359.99 degrees. If unknown, set to: UINT16_MAX |

Figure 6.4: MAVLink SYSTEM_TIME and GLOBAL_POSITION_INT MAVLink messages[21]

### 6.2.2 Transmitting the data from the Hex Cube Black

Firstly, through the Mission Planner, we configured the Telem2 port on the Hex Cube Black autopilot to use the MAVLink 2 protocol, as shown in Figure 6.5. Serial2 is another name for the Telem2 port.

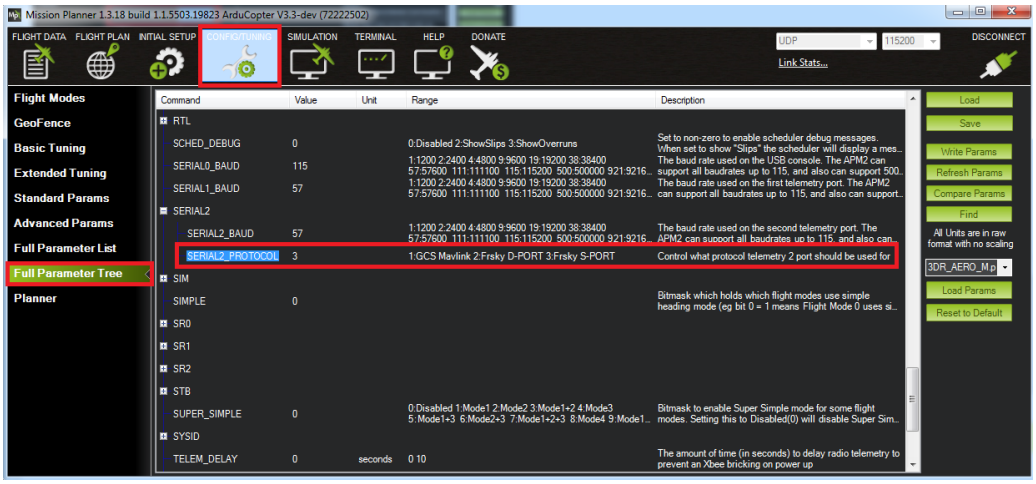



Figure 6.5: Configuration panel in Mission Planner, highlighted is the SERIAL2_PROTOCOL parameter[19]

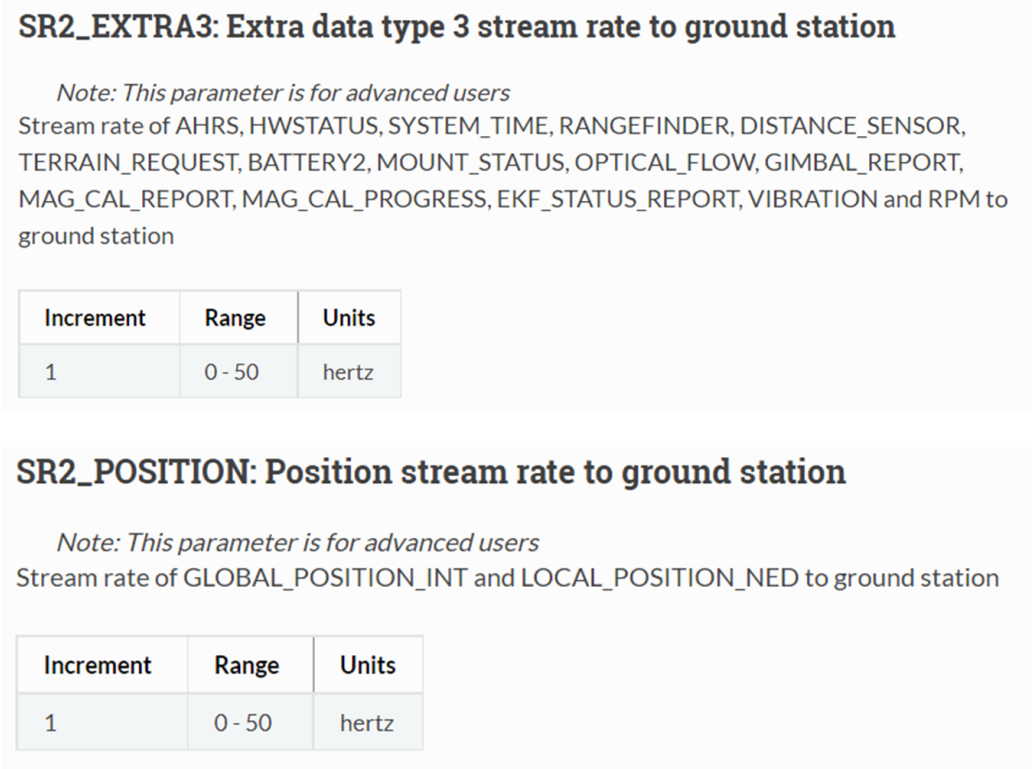
And we set the SERIAL2_BAUD parameter to 9, which sets the port baudrate to 9600.

The autopilot can send the data through the Telem2 port in 2 ways:

1. After receiving a REQUEST_DATA_STREAM MAVLink message with the ID of the requested message.
2. By setting appropriate parameters in the autopilot to have it continuously send groups of messages through the serial connection.

We used the second method since we always needed the same messages to be sent (SYSTEM_TIME and GLOBAL_POSITION_INT) and because this approach would remove the risk of lost packages. MAVLink, in

fact doesn't guarantee the reception of messages. At [22] can be found the list of all the ardupilot parameters. The ones we needed were the 'SR2_EXTRA3' and the 'SR2_POSITION'. Setting these parameters to a non zero value (possible values 0-10Hz) has the autopilot send different groups of messages. Figure 6.6 shows the messages in each group.



**SR2_EXTRA3: Extra data type 3 stream rate to ground station**

*Note: This parameter is for advanced users*

Stream rate of AHRS, HWSTATUS, SYSTEM_TIME, RANGEFINDER, DISTANCE_SENSOR, TERRAIN_REQUEST, BATTERY2, MOUNT_STATUS, OPTICAL_FLOW, GIMBAL_REPORT, MAG_CAL_REPORT, MAG_CAL_PROGRESS, EKF_STATUS_REPORT, VIBRATION and RPM to ground station

| Increment | Range | Units |
|---|---|---|
| 1 | 0 - 50 | hertz |

**SR2_POSITION: Position stream rate to ground station**

*Note: This parameter is for advanced users*

Stream rate of GLOBAL_POSITION_INT and LOCAL_POSITION_NED to ground station

| Increment | Range | Units |
|---|---|---|
| 1 | 0 - 50 | hertz |

Figure 6.6: SR2_EXTRA3 and SR2_POSITION data streams from ardupilot[22]

Through Mission Planner we set those parameters to 2Hz.

### 6.2.3 Receiving the data on the Raspberry Pi

The Telem2 port on the Hex Cube Black was connected to the *Universal Asynchronous Receiver-Transmitter* (UART) serial port on the Raspberry Pi 3b+ on pins GPIO 15 TxD (UART) and GPIO 16 RxD (UART). The serial connections was enabled in the Raspberry via `raspy-config` and was available on `/dev/ttyS0`. On the Raspberry Pi the script responsible for reading from the serial port and decoding the MAVLink messages is `mavlink_data.c`. It uses the official C library for MAVLink 2, c_library_v2, which contains the message definitions in XML format and helper functions to decode the messages. The code reads form the `/dev/ttyS0` serial port byte by byte; parses the message with the `mavlink\_parse\_char` function; checks the ID of the parsed message and if the ID is either 2 (SYSTEM_TIME) or 33 (GLOBAL_POSITION_INT) saves the information in the output csv file. The following code snippet contains the reading, parsing and decoding in the case of the SYSTEM_TIME message.

```
while(1) {
    if(flag)
        break;
    int n = read (fd, &byte, 1);
```

```
        if(n == -1) {
            fprintf (stderr, "error %d on read: %s", errno, portname, strerror (errno));
            return 1;
        }
        if (mavlink_parse_char(chan, byte, &msg, &status)) {
            printf("Received message with ID %d, sequence: %d from component %d of system %d\n", msg.ms
            switch(msg.msgid) {
                case MAVLINK_MSG_ID_SYSTEM_TIME: {
                    mavlink_msg_system_time_decode(&msg, &time_message);
                    mavlink_unix_time = time_message.time_unix_usec;
                    time_since_boot = time_message.time_boot_ms;
                    flag = true;
                }
            }
        }
    }
```

## 6.3 Code overview

The code is available at this Github repository[23]. The structure of the main script (`aria.py`) is the following:

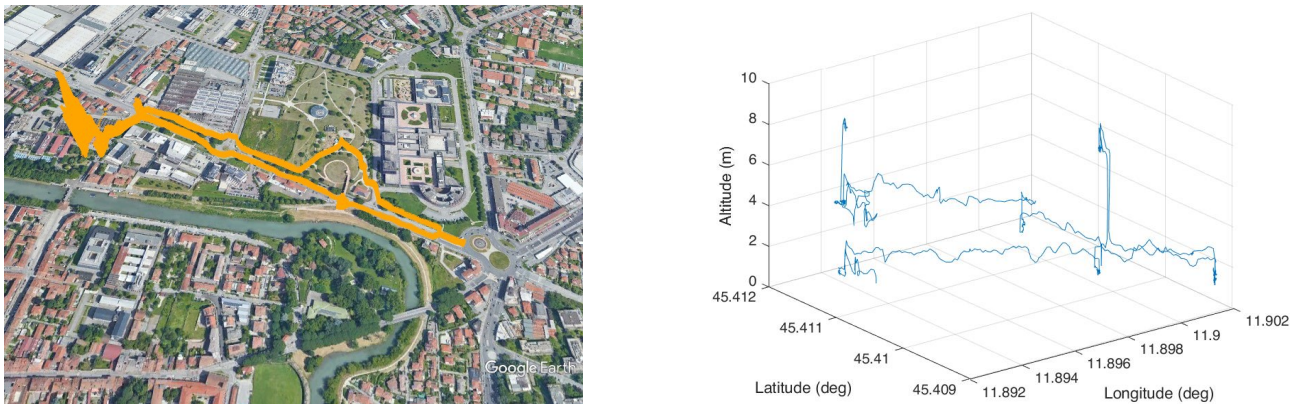
```
Open connection with the SDS011 sensor and the ADS1115 ADC
Call mavlink_data.c to check connection with the autopilot
Start an infinite loop:
    Call mavlink_data.c to get telemetry data
    Acquire data from particulate and gas sensors
    Process the data
    Write data to output csv files
    Real time plot of the data with pyplot
```

At every iteration of the loop both telemetry and sensors data are collected, so the rows of the csv files are time synchronized.

# Chapter 7

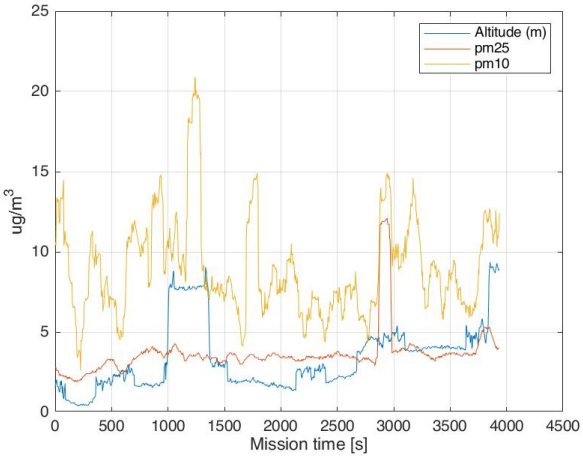
# ARIA: Field Test

A field test was performed in a trafficked area in Padova (Italy) in autumn 2021 (see Figure 7.1a), mainly to test the sensors. The drones were carried by hand. the Figures below contain the data collected by the two drones, the mission times are synchronized and the Arduino drone also has the data for the NO sensor. Firstly, the sensors and telemetry data are synchronized, as can be verified both by comparing the UNIX times in each row of the output csv files (see Section 6.1) and by comparison of the data from the Raspberry Pi and Arduino drones in Figure 7.2. The plot, in fact, shows that the

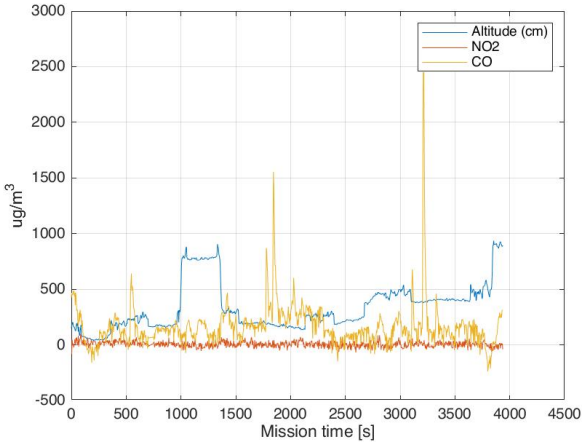



(a) (b)

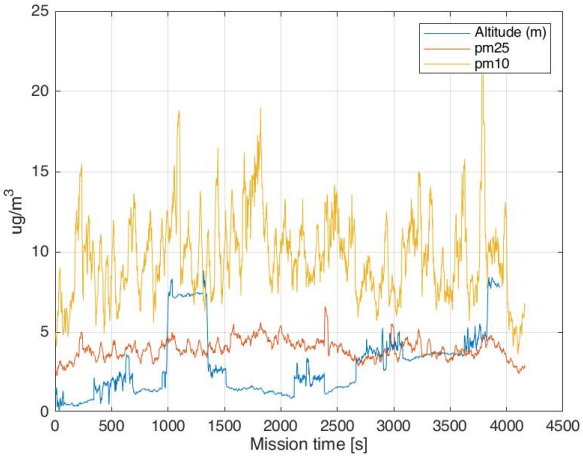
Figure 7.1: 3D map and 3D altitude plot



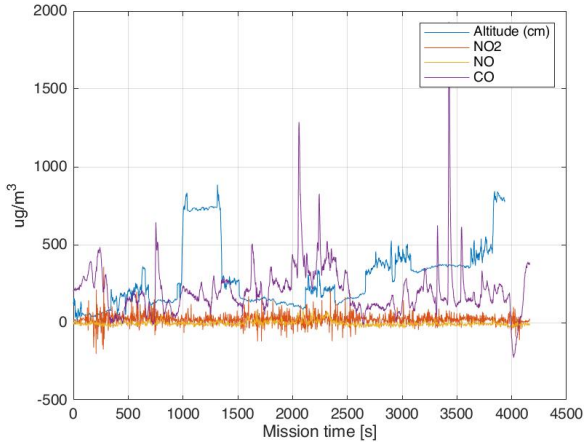

(a) Raspberry Pi drone PM sensor data




(b) Raspberry Pi drone gas sensor data

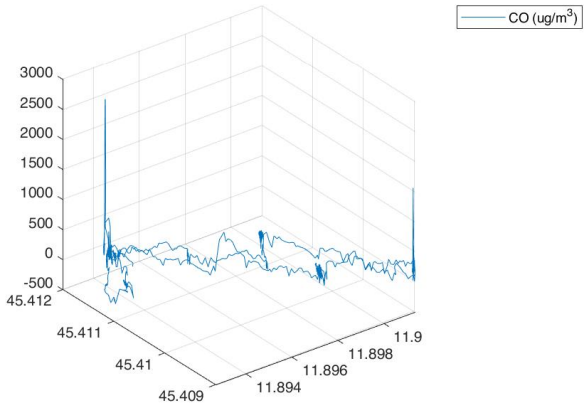



(c) Arduino drone PM sensor data

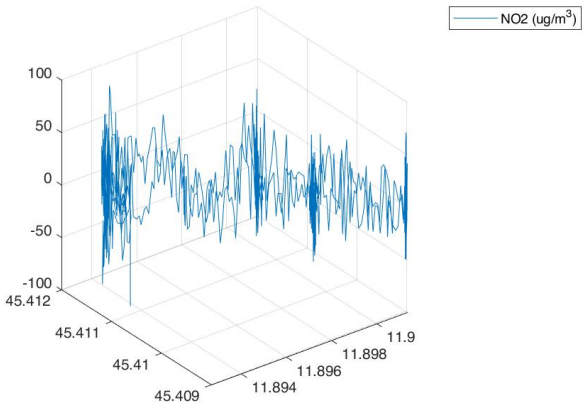



(d) Arduino drone gas sensor data

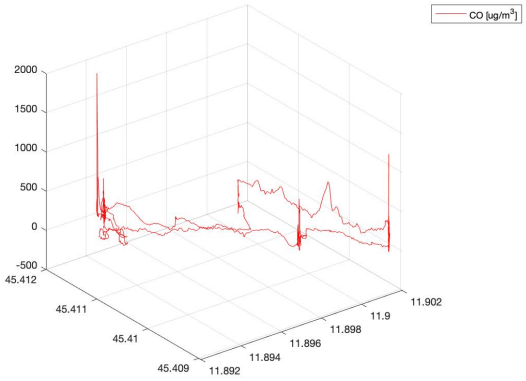
Figure 7.2: PM and gas sensors data from the Raspberry Pi drone (top two) and the Arduino drone (bottom two)



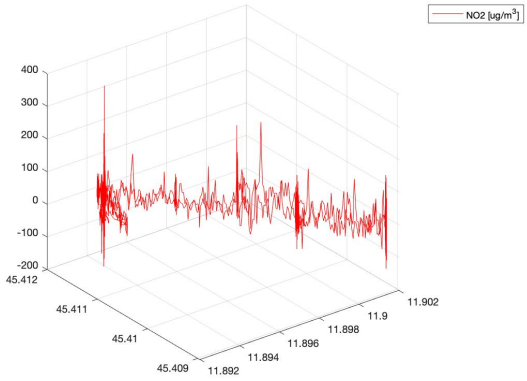

(a) Raspberry Pi $CO$



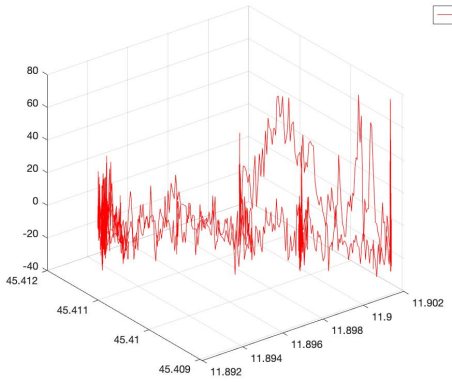

(b) Raspberry Pi $NO_2$




(c) Arduino $CO$




(d) Arduino $NO_2$




(e) Arduino $NO$

Figure 7.3: 3D plots of the gas sensors data, Raspberry Pi drone data is in blue (top two), Arduino drone data is in red (bottom three)

# Chapter 8

# Conclusions and future work

# Appendix A

# Alphasense gas sensors formulas

**Table 1 : Algorithms to correct the WE output for the effects of temperature**

| Algorithm | Equation | Notes |
|---|---|---|
| 1 | $WE_c = (WE_u - WE_e) - n_T * (AE_u - AE_e)$ | Subtraction of the electronic offsets from the raw $WE_u$ and $AE_u$ outputs then scales the net AE output with the $n_T$ factor. Gross under or over compensation can occur if ($AE_u - AE_e$) is of opposite sign to $n_T$, or if ($AE_u - AE_e$) is significantly smaller or larger than ($WE_u$-$WE_e$). Over compensation could lead to the final gas concentration appearing negative, or much higher compared to a reference value. |

(a) Algorithm 1

**Term definitions**

$WE_u$ = uncorrected raw WE output
$AE_u$ = uncorrected raw AE output
$WE_c$ = corrected WE output
$WE_e$ = WE electronic offset on the AFE or ISB
$AE_e$ = AE electronic offset on the AFE or ISB
$WE_o$ = WE sensor zero, i.e. the sensor WE output in zero air
$AE_o$ = AE sensor zero, i.e. the sensor AE output in zero air
$WE_T$ = Total WE zero offset
$AE_T$ = Total AE zero offset

$n_T$ = temperature dependent correction factor for algorithm 1, refer to Table 3 for values
$k_T$ = temperature dependent correction factor for algorithm 2, refer to Table 3 for values
$k'_T$ = temperature dependent correction factor for algorithm 3, refer to Table 3 for values
$k''_T$ = temperature dependent correction factor for algorithm 4, refer to Table 3 for values

(b) Symbols legend

5 Divide the final corrected WE result ($WE_c$) by the sensor's sensitivity to calculate the gas concentration. If desired, developers may wish to add a further refinement to the final gas concentration value by dividing the corrected WE output ($WE_c$) by a temperature corrected


Sensor Technology House, 300 Avenue West, Skyline 120, Great Notley. Essex.CM77 7AA. UK
Tel: +44 (0) 1376 556700 Fax: +44 (0) 1376 335899
Email: sensors@alphasense.com Web: www.alphasense.com



sensitivity instead of only the single ambient sensor sensitivity value provided. Each sensor type usually has a plot in its technical datasheet illustrating how the sensor sensitivity varies with temperature. Thus a complete temperature profile for the sensor type can be produced that corrects for background current changes (above) and sensitivity (or gain) changes due to temperature changes.

(c) The final calculation to obtain the gas concentration.

Figure A.1: The Alphasense gas sensors formula used in our tests, which is algorithm 1 from the AAN 803-05 from Alphasense.

# Bibliography


[1] Chih-Da Wu et al. "Mapping the vertical distribution of population and particulate air pollution in a near-highway urban neighborhood: Implications for exposure assessment". In: *Journal of exposure science & environmental epidemiology* 24.3 (2014), pp. 297–304.

[2] Hans Neuberger. "The vertical distribution of atmospheric properties and air pollution". In: *Air Repair* 3.3 (1954), pp. 205–210.

[3] Junhyun Park et al. "Low Cost Fine-Grained Air Quality Monitoring System Using LoRa WAN". In: *2019 International Conference on Information Networking (ICOIN)*. 2019, pp. 439–441. DOI: 10.1109/ICOIN.2019.8718193.

[4] Zixuan Bai et al. "Real-time Prediction for Fine-grained Air Quality Monitoring System with Asynchronous Sensing". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 7620–7624. DOI: 10.1109/ICASSP.2019.8682518.

[5] Gian Marco Bolla et al. "ARIA: Air Pollutants Monitoring Using UAVs". In: *2018 5th IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. 2018, pp. 225–229. DOI: 10.1109/MetroAeroSpace.2018.8453584.

[6] *Environmental Pollution Centers*. URL: https://www.environmentalpollutioncenters.org/.

[7] Josefa Wivou et al. "Air quality monitoring for sustainable systems via drone based technology". In: *2016 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*. 2016, pp. 1–5. DOI: 10.1109/ICIAFS.2016.7946542.

[8] *World Health Organization Air Quality Guidelines*. URL: https://www.who.int/news-room/feature-stories/detail/what-are-the-who-air-quality-guidelines.

[9] *Agenzia Regionale per la Prevenzione e Protezione Ambientale del Veneto*. URL: https://www.arpa.veneto.it/.

[10] Ron Williams et al. "Air sensor guidebook". In: *US Environmental Protection Agency* (2014).

[11] Wei Ying Yi et al. "A Survey of Wireless Sensor Network Based Air Pollution Monitoring Systems". In: *Sensors* 15.12 (2015), pp. 31392–31427. ISSN: 1424-8220. DOI: 10.3390/s151229859. URL: https://www.mdpi.com/1424-8220/15/12/29859.

[12] Orestis Evangelatos and José DP Rolim. "An Airborne Wireless Sensor Network for Ambient Air Pollution Monitoring". In: *SENSORNETS* (2015), pp. 231–239.

[13] Zhiwen Hu et al. “UAV Aided Aerial-Ground IoT for Air Quality Sensing in Smart City: Architecture, Technologies, and Implementation”. In: *IEEE Network* 33.2 (2019), pp. 14–22. DOI: 10.1109/MNET.2019.1800214.

[14] Qijun Gu and Chunrong Jia. “A Consumer UAV-based Air Quality Monitoring System for Smart Cities”. In: *2019 IEEE International Conference on Consumer Electronics (ICCE)*. 2019, pp. 1–6. DOI: 10.1109/ICCE.2019.8662050.

[15] *Alphasense CO-A4 Carbon Monoxide Sensor data sheet*. URL: https://www.alphasense.com/wp-content/uploads/2019/09/CO-A4.pdf.

[16] *Alphasense NO-A4 Nitric Oxide Sensor data sheet*. URL: https://www.alphasense.com/wp-content/uploads/2019/09/NO-A4.pdf.

[17] *Alphasense NO2-A43F Nitrogen Dioxide Sensor data sheet*. URL: https://www.alphasense.com/wp-content/uploads/2019/09/NO2-A43F.pdf.

[18] *Alphasense PID-AH2 Photo Ionisation Detector data sheet*. URL: https://www.alphasense.com/wp-content/uploads/2019/08/PID-AH2.pdf.

[19] *ArduPilot*. URL: https://ardupilot.org.

[20] *MAVLink*. URL: https://mavlink.io/en/.

[21] *Messages (common) - MAVLink developer guide*. URL: https://mavlink.io/en/messages/common.html.

[22] *Complete Parameters List - Copter Documentation*. URL: https://ardupilot.org/copter/docs/parameters.html#.

[23] *ARIA-project-unipd*. URL: https://github.com/GiacomoFavaron/ARIA-project-unipd.