A low-cost bicycle-based air quality monitoring system using a Raspberry Pi, with a pollution-avoidance routing application

Edward Jeffery

Master of Science
The University of Bath
October 2018

A low-cost bicycle-based air quality monitoring system using a Raspberry Pi, with a pollution-avoidance routing application

submitted by

Edward Jeffery

for the degree of Master of Science of the

University of Bath

October 2018

DECLARATION

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Master of Science in the Department of Computer Science. No

portion of the work in this thesis has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signature of Author .....

Edward Jeffery

Abstract

Air pollution is a major health problem, causing millions of deaths each year, as well as many other non-fatal diseases. In order for authorities to be able to mitigate air pollution effectively, they need access to high-resolution air quality data. Currently, air quality monitoring in the UK depends on fixed monitoring stations, which offer poor resolution, both spatially and temporally. In this paper, we design and construct an air quality monitoring device that attaches to a bicycle, thus providing a mobile platform for data collection. We show that it is robust and effective and collecting air quality data. We also show the potential improvements in spatial and temporal resolution compared to fixed monitoring stations, as well as its ability to rapidly be deployed to major events for air quality monitoring. Furthermore, we present an innovative routing algorithm that finds 'health-optimal' routes, decreasing exposure by X% on average compared to the shortest routes.

Acknowledgements

I would like to thank my research project supervisor, Dr. Julian Padget, for his guidance and assistance throughout this project and during the course.

I would also like to thank my parents, David and Melanie Jeffery, and my girlfriend, Isabel Cairns, for their ongoing support throughout the writing of this thesis.

Martin Fullick (Department of Physics) was a great help with the 3D printing and soldering. His knowledge was invaluable.

Thank you to my trusty bicycle, which only suffered 3 punctures throughout the whole project, despite the hundreds of potholes that litter Bath's roads.

R.I.P. to my lungs and eyes, which suffered greatly throughout the data collection on Bath's roads.

Contents

List of Figures

List of Tables

List of Abbreviations

ADC Analog-to-digital converter

ANN Artificial neural network

API Application programming interface

AURN Automatic Urban and Rural Network

BANES Bath and North East Somerset Council

BC Black carbon

CO Carbon monoxide

DAQI Daily Air Quality Index

Data Quality Objective DQO

DEFRA Department for Environmental and Rural Affairs

EEA European Environment Agency

FTP File Transfer Protocol

GIS Geographical-information system

GPIO General Purpose In/Out

GPS Global positioning system

HAT Hardware-on-top

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

LED Light emitting diode

LMIC Low and middle-income countries

LPO Low Pulse Occupancy

LUR Land-use regression

MAD Median absolute deviation

MCU Microcontroller unit

MSB Most Significant Bit

NOx Nitrogen oxides

NTP Network Time Protocol

O_3 Ozone

PM Particulate matter

ppb Parts-per-billion

REST Representational State Transfer

RPi Raspberry Pi

SBC Single-board computer

SPI Serial Peripheral Interface

UFP Ultra-fine particle

VCC Voltage common collector

VSN Vehicle sensor network

WHO World Health Organisation

Chapter 1

Introduction

The World Health Organisation (WHO) estimates that 90% of the world's population lives in places that do not comply with the WHO Air Quality Guidelines and 3 million people each year die from breathing polluted air (WHO, 2016). Disconcertingly, this is considered a conservative measure by the WHO because it focuses on particulates and excludes health impacts of other pollutants such as nitrogen oxides (NOx) and ozone (O_3). Air pollution is also not just a problem in the most industrialised countries, but is a widespread problem around the world with 87% of these deaths occurring in low and middle-income countries (LMICs). Besides the aforementioned fatalities, air pollution has a number of other serious consequences, such as cardiorespiratory and cardiovascular disease (Pope et al., 1995; Dockery et al., 1993; Pope et al., 2002).

In addition to the health problems associated with air pollution, air quality monitoring equipment is relatively expensive, with professional sensing equipment reaching into the thousands of pounds. Within the last ten years, low-cost sensors have been developed that are now widely available as a cheap alternative to their more expensive counterparts (Clements et al., 2017). Coupled with the decline in both the cost of computers and their size, creating low-cost monitoring equipment for air quality monitoring purposes has become a more realistic opportunity. Attempts have been made at creating such devices, but studies focusing on stationary devices require a large number to be installed to get distributed readings in a given location because of the large spatial variability of these pollutants in urban environments. In addition, many are indoor-based devices and thus do not

measure ambient (i.e. outdoor) air quality. An alternative is to use a mobile platform, which can assess the spatial variability using far fewer sensing devices and in a constrained time-frame (Hagler, Thoma and Baldauf, 2010; Wallace et al., 2009; Westerdahl et al., 2005; Weijers et al., 2004; Adams et al., 2012; Elen et al., 2013). This introduces the concept of 'runs', whereby the vehicle carrying the sensing device takes repeated measurements over the entire area at different days and/or different times of one day (Peters et al., 2013).

A further issue hampering the development of these devices is the inaccuracy of low-cost sensors. They are often sensitive to environmental factors, such as temperature, and can have cross-sensitivities with other pollutants. As a result, calibration methods have been developed that implement multi-variate linear regression and different machine learning techniques to increase the accuracy of these sensors (Spinelle et al., 2015, 2017; Zimmerman et al., 2018).

Many different vehicles have been used previously when monitoring air quality. A similar study was conducted using a vehicular sensor network on public buses (Re, Peri and Vassallo, 2014). A different study based the monitoring on public trams (Hagemann et al., 2014), indicating that public transport is a viable way of making the sensing mobile. Another study used a tricycle-based vehicle (Jabbar et al., 2017) and standard bicycles have also been used (Elen et al., 2013; Peters et al., 2013). Alternatively, other options include: fuel-based cars (Apte et al., 2017; Devarakonda et al., 2013), electric cars (Hagler, Thoma and Baldauf, 2010), Post Office vehicles, couriers, and other public transport. Those papers in which a non-polluting vehicle was used stress the advantage of using a vehicle that does not affect the measurements with its own pollution. Furthermore, bicycles can be used both on the road as well as in areas that typical traffic cannot go. For these reasons coupled with the resource availabilities to the project, a bicycle will be used to host the monitoring device.

There are a number of potential problems with building a portable sensing device. The device needs to be supplied with consistent and reliable power, and it needs to be robust to varying environmental conditions. It also needs to be small and light enough to attach onto a vehicle without affecting its usability. Additionally, the device needs to be able to remotely send its readings for processing, storage and display.

If the project is successful and a reliable air-quality map of Bath can be gen-

erated, there is the potential that routes identified to have poor air quality can be avoided, which may lead to cardiopulmonary/cardiorespiratory health benefits. In fact, Google (2017a) are trialling an air quality monitoring project in California with the view to incorporate this data into their Google Maps product. Rather than using the typical path-finding algorithms, the pollution level can be incorporated into the cost function in order to re-direct the route away from areas of high pollution. In addition to Google's efforts, an air quality monitoring project in Zurich found that taking the health-optimal path instead of the shortest path could lead to a 7.1% reduction in exposure (Hasenfratz et al., 2015). Furthermore, if air pollution exposure is found to be high for cyclists, then the council could build alternative cycle paths that distance cyclists from the pollution and help to reduce exposure (Schepers et al., 2015). There is also an educational aspect from presenting air quality data to the public because citizens can learn more about air pollution and how it affects them.

As a proposed solution to the problems detailed above, this project aims to design and create a low-cost, portable air quality monitoring device that can be attached to a vehicle and collect measurements whilst the vehicle is moving around the city of Bath. Making the device low-cost is important for air quality monitoring becoming ubiquitous in the future and the device being mobile is also important in order to improve the spatial resolution of the air quality monitoring in comparison to current fixed stations. Furthermore, this project will also seek to assess the feasibility of using low-cost sensors to accurately measure air pollution and attempt to enhance the calibration accuracy of the low-cost sensors over time using either regression or machine learning techniques. In addition, if the device is a success, there is also the possibility of deploying multiple devices so that a larger sample set and greater geographical coverage can be obtained compared to if a singular device was used. Finally, the data will be used to generate pollution maps of Bath, which should inform the public about the air quality in their area.

In terms of more technical objectives, the air quality monitoring system as a whole should be robust. The data flow from collection through to display will ideally be smooth and reliable, and data processing and cleaning will form part of this process.

In this paper, we present a bicycle-based air quality monitoring device. In Chapter 2, we cover the literature on air pollution and existing air quality mon-

itoring platforms. In Chapter 3, we describe how the device was designed, from both a physical and software standpoint. Chapter 4 then details how the raw sensor outputs were read and converted to meaningful readings. Chapter 5 deals with the experiment setup and how we aim to answer our research questions arising from our review of the literature. Next, we detail our results in Chapter 6 and discuss our findings in Chapter 7. Finally, Chapter 8 details our conclusions and recommendations for future work.

Chapter 2

Related work

This research project to design and build a low-cost air quality monitoring system covers a wide number of topics. Given the relative infancy of low-cost sensors and computing, making such a system has only been made possible in recent years and, as such, the literature needs thorough investigation to understand where the current implementations can be improved to make widespread air quality monitoring a more realistic possibility.

Firstly, the health impacts of air pollution are considered, outlining why being able to monitor air pollution easily is a major concern. The various measures of air quality are then detailed in order to establish an understanding of what are the most important pollutants to measure. Next, previous monitoring devices are listed and the more relevant studies are evaluated to identify where novel research is needed and where improvements on past studies can made. Potential candidates for the base computer of the proposed air quality monitoring device are then compared, before power solutions to the problem of portably powering the device are then considered. The specific drawbacks of using low-cost sensors are then detailed, as well as a discussion of techniques in the literature for improving them. Finally, methods for mapping air quality data and monitoring the data quality are analysed.

2.1 Air pollution

Air pollution is an almost invisible threat to the population. It contaminates the air that we breathe and can cause adverse health effects. This section details these health effects, as well as detailing exactly why there is a need for monitoring air pollution.

2.1.1 Health impact of air pollution

There is a known negative relationship between air pollution and health. Dockery et al. (1993) conducted one of the first studies that demonstrated an association between air pollution and increased mortality rates whilst controlling for smoking-related factors. A similar study by Pope et al. (1995) examined sulphur dioxide and particulate air pollution as predictors of mortality in the United States. When also controlling for smoking, once again they found that there was an increase in mortality rates associated with levels of pollution commonly found in cities. The particulate air pollution was also associated with an increase in lung cancer rates. A later study by Pope et al. (2002) expanded on the aforementioned study by doubling the follow-up time, expanding the exposure data and controlling for more variables. The authors concluded that combustion-related fine particulate air pollution increased the risk of cardiopulmonary and lung cancer mortality. More recent studies have also confirmed these results (Anderson, Thundiyil and Stolbach, 2012; Beelen et al., 2014).

A less serious, but still widespread, health effect of air pollution is exacerbation of asthma symptoms, particularly in young children (Jerrett et al., 2008). Additionally, a positive relationship has been found for new-onset asthma rates, although the strength of the evidence is variable (Guarnieri and Balmes, 2014).

A recent review of studies assessing the long-term link between nitrogen dioxide (NO_2) exposure and mortality found sufficient evidence to suggest the effect on mortality was as great as for particulates (Faustini, Rapp and Forastiere, 2014). Furthermore, although studies on the effect of ozone (O_3) air pollution are sparse, Turner et al. (2016) found that long-term exposure to ambient ozone contributes to an increased risk of respiratory and circulatory mortality.

2.1.2 The need for monitoring air pollution

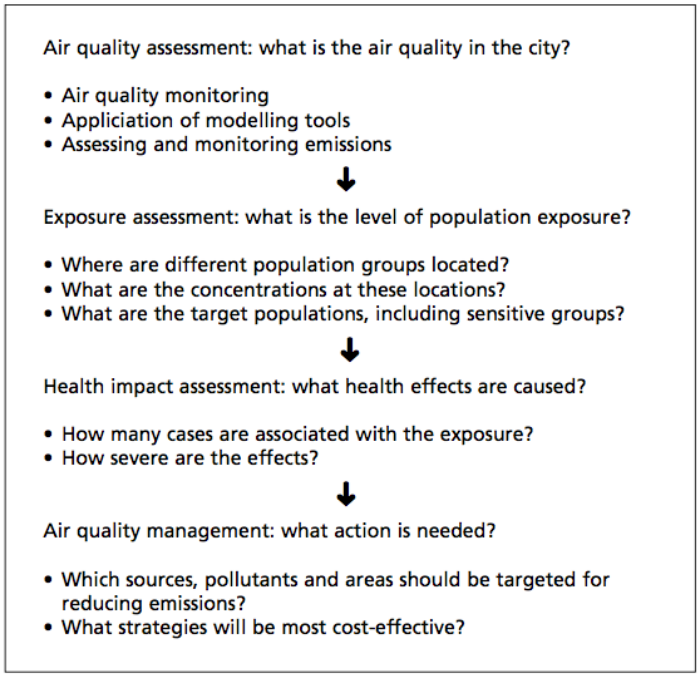


Figure 2-1: Rationale for air quality monitoring (WHO, 2002)

Air pollution is formed up of gases and microscopic particles, which are often invisible to the naked eye. Therefore, in order for any pollution reduction strategies to be effective, there needs to be a method of measuring and monitoring air pollution so that problem areas can be identified.

A report by the WHO (2002) looked at the importance of monitoring air quality for health impact assessment purposes. At the root of the rationale is identifying what the air quality is like in a given location. Second, the report details how the nature of population exposure must first be understood in the chain. Third, the population exposure needs to be assessed to understand the health impact. Finally, the health impact then should directly impact plans for air quality management and improving public health. This rationale is detailed is Figure 2-1.

Furthermore, the main pollutants that need to be monitored must be identified. Section 2.2 discusses these.

Table 2.1: UK and EEA air quality guidelines

	Limit ($\mu g/m^3$)				
Gas/particulate	15-minute	1-hour	8-hour	24-hour	Annual
Fine particulate matter ($PM_{2.5}$)	–	–	–	50	40$^{\diamond}$
Particulate matter (PM_{10})	–	–	–	–	25
Nitrogen dioxide (NO_2)	–	200†	–	–	40
Ozone (O_3)	–	–	100†	–	–
Sulphur dioxide (SO_2)	266$^{\diamond}$	350‡	–	125*	–

*Not to be exceeded more than 3 times per year
†Not to be exceeded more than 10 times per year
‡Not to be exceeded more than 24 times per year
$^{\diamond}$Not to be exceeded more than 35 times per year

2.2 Measures of air quality

Many governments have central bodies that handle the measuring of air quality and the publishing of the readings so that the public can view them. In the UK, the Department for Environmental and Rural Affairs (DEFRA) are responsible for this and they release a yearly report on air quality statistics (DEFRA, 2017a), as well as daily and weekly readings on their official website (DEFRA, 2018b). The data used for these are collected using the Automatic Urban and Rural Network (AURN) and the statistics focus on five main pollutants, detailed below in Table 2.1 These measurements make up the Daily Air Quality Index (DAQI), which is used to inform the public about the current or forecast risk of air pollution (DEFRA, 2018a). Additionally, the European Environment Agency (2018) recently launched its own air quality index, which measures the same five pollutants. Both the UK and the EEA have air quality guidelines that specify a minimum air quality level that must not be exceeded. These are also detailed below in the table, with the more stringent limit taken in each case.

In this study, we focus on particulates, nitrogen oxides and ozone. Further information about these pollutants is given below.

2.2.1 Nitrogen oxides

Nitrogen oxides (NOx) is a collective term for a group of reactive nitrogen-based substances that includes nitrogen monoxide (NO) and nitrogen dioxide (NO_2). Research generally focuses on nitrogen dioxide because it one of the commonly regulated air quality pollutants (Brook et al., 2004). In addition to this, NO rapidly oxidises with oxygen in the air to form NO_2 and so is not as prevalent in the environment. It also plays an important role in the formation of ozone (O_3). One of the main sources of nitrogen oxides is the combustion of fossil fuels, which is typically found in petrol or diesel vehicles. Long-term exposure to NO_2 is associated with an increase in mortality rates (Faustini, Rapp and Forastiere, 2014).

2.2.2 Ozone

Ozone (O_3) is another pungent gas that is highly reactive. More specifically, ozone that is classed as a pollutant is known as 'tropospheric ozone' as it is found near ground-level. Low-level exposure is widespread as it is formed naturally in the atmosphere. In addition to this, the action of solar UV radiation on NOx leads to the formation of O_3, meaning concentrations are generally higher on sunnier days (Brook et al., 2004). O_3 is a known pulmonary irritant (Ebi and McGregor, 2008) and long-term exposure to ambient ozone is related with an increased risk of respiratory and circulatory mortality (Turner et al., 2016).

2.2.3 Particulates

Particulates or particulate matter (PM) are mixtures of solid matter and liquid matter that are suspended in the air. They come in various sizes, of which the standard measures are PM_1, $PM_{2.5}$ and PM_{10}. These refer to the measured quantity of particles with size $<1\,\mu m$, $<2.5\,\mu m$ and $<10\,\mu m$, respectively. PM is created in two ways: (i) by direct emission, or (ii) by the physicochemical transformation of gases (Brook et al., 2004). Inhalation of $PM_{2.5}$ and PM_{10} is associated with many adverse health outcomes (Pope et al., 1995, 2002; Anderson, Thundiyil and Stolbach, 2012; Beelen et al., 2014).

2.3 Monitoring urban air quality

Given the serious health implications of air pollution detailed in Section 2.1.1 and the prevalence of minimum air quality regulations around the world, air quality monitoring is of utmost importance, particularly in urban environments where pollution is often highest. To get a distributed measurement of air pollution in an area, multiple devices are often deployed. The geographical distance between these devices affects the *spatial* resolution of the air quality data, whilst the repeated collection of measurements from the same location affects the *temporal* resolution.

2.3.1 Previous monitoring devices

There are a large number of different air quality monitoring devices detailed in the literature. They use a variety of different computers upon which the sensors are based, as well as containing sensors for measuring a number of different gases and particulates. These devices are either deployed in a fixed location, or they are attached to different vehicles for taking non-static measurements. Table 2.2 provides a comprehensive overview of air quality monitoring devices in the literature, detailing the vehicle used, which gases and/or particulates were measured and what computer the device was based on. Particularly relevant studies in this table are discussed below.

One example of an air quality monitoring device is an Arduino-based device deployed on a public transport bus fleet (Re, Peri and Vassallo, 2014). They built a network of these devices, called a Vehicular Sensor Network (VSN). Each device communicated its air quality measurements to fixed access points via radio, and then these forward the data on to a central server. The authors found that using only a few vehicles with these monitoring devices attached, they could monitor a whole city of size $160km^2$. This paper is interesting in the fact that it presents the current air quality monitoring system of the analysed city, before explaining how a network of mobile sensors could improve the monitoring.

Other public transport has been used to attach air quality monitoring devices. Hasenfratz et al. (2015) attached monitoring devices to trams in Zurich, Switzerland, and collected measurements over a two-year period. By using the tram network, they were able to cover a large urban area of 100km^2 on a reg-

ular basis. For the base computer of their device they used a Gumstix, which is similar to a Raspberry Pi. The device itself stored measurements in a local database and transmitted them in real-time over a cellular network, before then deleting stored readings once receipt was confirmed. Again, the paper highlighted the advantage of this mobile system compared to a static government monitoring station. Furthermore, their novel contribution was to derive a cost function for all street segments of Zurich's road network, which was then incorporated into a route finding algorithm to compute the health-optimal path. This was then compared to the route generated without the inclusion of this cost function and the authors found that taking the health-optimal path led to a 7.1% reduction in particulate pollution exposure. Hagemann et al. (2014) also used trams to assess spatial variability of particle number concentrations and NOx.

Another study used a device attached to a bicycle, named Aeroflex (Elen et al., 2013). The focus was on high adaptability and ease of use since everybody is 'able to ride a bike', thus enabling any volunteer member of the public to contribute to air quality monitoring. This ease of use was not only for the bicycle, but also for the software that collected, sent and displayed the data. All of the air quality monitoring equipment was attached to the bicycle (see Figure 2-2), including a robust data transmission system that could provide real time measurements on air quality. The Aeroflex took measurements every second and was ridden around a set route multiple times in order to provide a high spatial resolution and to ensure good temporal resolution. The success of the project was demonstrated by city authorities using it in a number of cities in Belgium, including Antwerp, Ghent and Brussels. Whilst the novel use of a bicycle to monitor air quality is a major advantage of this study, it only focused on measuring particulates, carbon monoxide and black carbon. Out of these, only particulates are part of the indexes used by the UK and the EEA, which questions the device's usefulness in helping the city authorities in monitoring air quality so that air pollution limits are not broken. This identifies a gap in the literature for a bicycle-based monitoring system that measures more of the gases and particulates detailed in Section 2.2.

Another interesting study was conducted by Apte et al. (2017) using Google Street View cars in California. The study covered 24,000km within a 30km^2 area, meaning it had much higher spatial and temporal resolutions than previous mobile monitoring studies. By also incorporating this data with Google Street

Figure 2-2: Aeroflex air quality monitoring platform developed by Elen et al. (2013)

View's imagery, it was possible to analyse 'hotspots' and suggest plausible causes of the problem, thus making the dataset richer. For example, a metal recycling business was identified to be the cause of a hotspot for black carbon, nitrogen oxide and nitrogen dioxide. Having this level of detail would greatly help in informing council decision-making relating to air pollution. The authors also note that having routine availability of high-resolution air quality data in cities could make it possible to alter personal behaviour, much like real-time traffic data currently affects driving decisions. However, the study used industry-grade monitoring equipment instead of a low-cost device. Whilst this is feasible for a company such as Google, this is not conducive to making low-cost monitoring widespread.

Looking at the air quality monitoring literature as a whole, a number of areas remain unaddressed. Firstly, no study measure all of pollutants detailed in Section 2.2 that form UK and EU legislation. Whilst this may be because it is difficult to fit that number of sensors into one device, if mobile air quality monitoring in the future is to be a viable option in the UK, this needs to be addressed. Secondly, the capabilities of using a sole Raspberry Pi device for air quality monitoring purposes needs further investigation as it is often used in conjunction with another computer (see Table 2.2). Thirdly, only three of the studies detailed in Table 2.2 used bicycles, which are ubiquitous, non-polluting vehicles that would appear

to be an ideal candidate vehicle for mobile air quality monitoring. Finally, in all of these three cases, the air quality monitoring device appeared bulky and cumbersome (see Figure 2-2 for an example). Therefore, it would be interesting to see if all of the sensing capabilities could be incorporated into a small device.

Table 2.2: List of air quality monitoring devices in the literature

Study	Vehicle type	Gas/particulates measured	Base computer
Jabbar et al. (2017)	Tricycle	$PM_{2.5}$	RPi
Alvear et al. (2016)	Bicycle	CO_2, O_3	RPi & Waspmote
Apte et al. (2017)	Car	BC, NO, NO_2	Aclima Ei
Devarakonda et al. (2013)	Bus & Car	CO, PM	Arduino
Elen et al. (2013)	Bicycle	PM_1, $PM_{2.5}$, PM_{10}, BC, CO	Netbook
Ferdoush and Li (2014)	Stationary	Temperature, humidity	RPi & Arduino
Hagemann et al. (2014)	Tram	CO, CO_2, NO, NOx, O_3	Custom
Hagler, Thoma and Baldauf (2010)	Electric car	CO, UFPs	N/A
Hasenfratz et al. (2015)	Tram	O_3, CO, NO_2, UFPs	Gumstix
Hoang et al. (2013)	Motorcycle	CO	Waspmote
Peters et al. (2013)	Bicycle	PM_{10}, UFP	N/A
Piedrahita et al. (2014)	Person (wearable)	CO, CO_2, NO_2, O_3	Arduino
Re, Peri and Vassallo (2014)	Bus	NO_2, CO_2, CO, O_3	RPi & Arduino
Sun et al. (2016)	Stationary	CO, NO_2, O_3, $PM_{2.5}$	Arduino
Wallace et al. (2009)	Van	NOx, SO_2	N/A
Weijers et al. (2004)	Van	PM_1, $PM_{>1}$	N/A
Westerdahl et al. (2005)	Car	CO, CO_2, $PM_{2.5}$, NO, NO_2	N/A
Wong, Chua and Li (2009)	Car	CO, NOx	Renesas MCU

2.3.2 Spatial variability

A number of studies have shown that air pollution in cities can vary over extremely short distance (< 100 m) [REFs]. Therefore, air pollution measurements from a fixed monitoring station are likely only representative of a small radius around the site. This acts as a trade-off with the improved accuracy that fixed monitoring stations provide. Furthermore, with these stations typically located in areas of high pollution that the local authorities wish to monitor, the measurements may provide an inflated estimate if taken to be representative of a large area.

It has also been noted that the presence of unbroken rows of tall buildings on either side of a road can trap air pollution. These types of roads have been named 'street canyons'. In addition, some evidence suggests that the level of pollution behind these buildings can drop to background levels, indicating the dramatic variability in pollution that can occur in a city.

2.3.3 Spatial resolution

When monitoring air quality, a higher spatial resolution is considered to be better because air quality can be mapped in greater detail and exact problem areas identified more easily.

Air pollution is typically measured mainly using a network of fixed monitoring stations (Adams et al., 2012). Since they are fixed, they can be attached to a consistent power source and can include an array of measuring equipment that can measure a wide variety of pollutants with a level of accuracy that is often not possible on smaller, portable devices. However, these stations are likely to be placed in locations with high levels of air pollution (Kanaroglou et al., 2005) so that accurate measurements can be taken to ensure pollution regulation limits are not broken. The high cost and maintenance of these stations also means that only a low number can be built and deployed (Devarakonda et al., 2013), which leads to two possibilities arising: (i) a smaller area is covered with higher spatial resolution, or (ii) a larger area is covered with lower spatial resolution. In either case, the spatial resolution remains poor due to the low overall number of deployed devices.

Conversely, a monitoring device attached to a moving vehicle – i.e. a *mobile* device – can cover a larger area for a lower cost since a much smaller number of

devices needs to be used (Wallace et al., 2009). The area being monitored is also flexible as a moving device does not have the same restrictions as a fixed one. Moreover, setting a suitably short interval for taking air quality measurements enables the moving device to have a greater spatial resolution as many measurements can be taken in close proximity to one another. For example, Apte et al. (2017) took measurements every 30 metres and found using a mobile system led to a 10^4 to 10^5 times greater – i.e. finer – spatial resolution compared to fixed monitoring stations. These improvements do come at a cost, however. Temporal resolution is reduced because any given location will only have multiple measurements once the moving device visits that exact location again (Wong, Chua and Li, 2009). One solution to this problem is to increase the number of vehicles deployed with the monitoring device attached.

The apparent advantages of mobile air quality monitoring over static monitoring and its increasing prevalence in the literature suggest it is more suitable for this project. In addition, Kuhlbusch et al. (2014), in a collective statement about the future of European air quality monitoring, recommend mobile devices for 'the collection of highly spatially and temporally resolved data'.

2.3.4 Temporal resolution

Much of the air quality literature relating to the resolution of measurements focuses on the spatial resolution. However, the temporal resolution is also important as higher temporal resolution allows one to better understand how the pollution in an area changes over time.

2.4 Comparison of single-board computers

Whilst this section does not intend to provide a comprehensive comparison of single-board computers (SBCs) for use in this project (this would be a significant project itself), it is worth considering the relative advantages and disadvantages of various SBCs.

The decreasing cost and size of computers over time has created a market for low-cost, single-board computers that are compact, but have significant computing capabilities. There are many different makes of SBCs available, but the most

commonly used SBCs in the air quality monitoring literature are compared in this section. Whilst an Arduino is technically a microcontroller unit (MCU), its prevalence in the literature means it should also be included in this comparison.

2.4.1 Raspberry Pi (RPi)

Due to the small size and low-cost of Raspberry Pis, they are viewed as an ideal computer to run an air quality monitoring device. These SBCs do not come with any sensing capabilities built-in, meaning either sensors need to be attached or a separate circuit board containing the sensors needs to be connected to the RPi. Table 2.3 shows the details of four popular RPi models. Only the RPi 3 and RPi Zero W have Wi-Fi and Bluetooth capabilities, which would be useful in this kind of project for transferring data from the monitoring device. In addition, the RPi 3 has greater computational power, but requires substantially more power and costs over three times as much. The RPi Model A+ and RPi 2 have no clear merit over the other two RPis, suggesting using a RPi 3 or RPi Zero W would be sensible.

Raspberry Pis have been used in air quality research previously (Ibrahim et al., 2015; Balasubramaniyan and Manivannan, 2016; Rahman et al., 2017; Thorpe, 2017; Alkandari and Moein, 2018), with one study attaching them to public buses (Re, Peri and Vassallo, 2014) and another using them as part of tricycle-based mobile system (Jabbar et al., 2017). The use of Raspberry Pis in these studies confirms its suitability for air quality monitoring.

There are a variety of Raspberry Pi models available, with an apparent trade-off between size and power consumption and computing power. Thorpe (2017) used both the Raspberry Pi 3 Model B and the Raspberry Pi Zero W for particulate sensing devices and found the former to have a much more reliable WiFi signal. However, the distance between devices was up to 30 metres and the Raspberry Pi used in this project will not be subjected to sending data over such distances given the whole system will be installed on one vehicle. This suggests the RPi Zero W may be suitable.

One issue that Raspberry Pis have is that they do not have an onboard clock and so the system time is the time since the last boot. To alleviate this problem, a time source is needed to accompany the RPi.

Furthermore, the type of sensors being used in this project output an analog signal, which the RPi cannot read directly. As a result, an analog-to-digital converter (ADC) would need to be fitted to the sensor board.

2.4.2 Arduino

An Arduino is a single-board microcontroller motherboard which has been widely used in the environmental monitoring literature (Re, Peri and Vassallo, 2014; Devarakonda et al., 2013; Balasubramaniyan and Manivannan, 2016; Sun et al., 2016; Ferdoush and Li, 2014; Lee et al., 2014; Alvear et al., 2016; Fuertes et al., 2016; Piedrahita et al., 2014; Abraham and Li, 2014). The hardware is open source with a large community of support and the software is open source, too. Since it is a microcontroller, it can only run one program at a time. Whilst its CPU speed from Table 2.3 seems low, a microcontroller is not comparable to the SBCs in this respect and so this type of comparison should be avoided. Despite this, a microcontroller is good for simple, repetitive tasks and as a result, is considered to have relatively fast computation. Moreover, Arduinos read analogue inputs, which is beneficial since sensors typically output an analogue signal. However, an Arduino is unsuitable for multiple, complex tasks that require more than one program to be run. Furthermore, an Arduino does not come with built-in Wi-Fi or Bluetooth (see Table 2.3), meaning an additional dongle would be needed.

2.4.3 BeagleBone

The BeagleBone is a low-cost, community supported SBC that runs Linux. It has been used successfully in air quality monitoring studies, including a study in which it was part of a *real time* air quality monitoring system (Min, Forys and Schmid, 2014). The BeagleBone passed measurements on to a web interface using a Wi-Fi connection, but had an additional local SQLite database as backup in order to mitigate variations in Wi-Fi signal. The authors also connected an LCD screen to the BeagleBone in order to aid with the usability of the device. In a different study by Desai and Alex (2017), the BeagleBone stored comma-separated value (CSV) files containing the day's measurements, which were then uploaded to a cloud-based database at the end of the day.

The device itself is almost twice the price of the second-most expensive SBC

in Table 2.3 and has the highest power requirement. It also only has the same computational power as the RPi zero W, which is a smaller and cheaper device. However, it does have built-in Wi-Fi and Bluetooth, which would be beneficial.

2.4.4 Comparison

Table 2.3: Comparison of SBCs

Model	Recommended / min. current[4]	CPU speed	RAM	Wi-Fi built-in	Blue–tooth	Price
RPi Model A+	700mA/180mA	700MHz	512MB	No	No	£20.00[1]
RPi 2 Model B	1.8A/350mA	900MHz (QC)	1GB	No	No	£34.00[1]
RPi 3 Model B	2.5A/400mA	1.2GHz (QC)	1GB	Yes	Yes	£32.00[1]
RPi Zero W	1.2A/150mA	1GHz	512MB	Yes	Yes	£9.16[1]
Arduino Uno	100mA/50mA	16MHz	N/A	No	No	£16.64[2]
BeagleBone Black W	1.2A/500mA	1GHz	512MB	Yes	Yes	£62.24[3]

[1]Source: https://thepihut.com/collections/raspberry-pi
[2]Source: https://store.arduino.cc/
[3]Source: https://beagleboard.org/black-wireless
[4]All require 5V, except the Arduino Uno which requires 7-12V.
(QC) indicates that the CPU is quad-core.

Given the greater usage of the Raspberry Pi and Arduino in the literature (see Table 2.2), it would be sensible to also use one of these devices for this project. The Arduino has the advantage of a lower power consumption and can read signals from the sensors without the need of an ADC chip. However, it is a microcontroller and so does not have the same computational capabilities or flexibility as the RPi. Comparing all of the different RPi models, the RPi 3 and the RPi Zero W seem the most sensible options. They both have built-in Wi-Fi and Bluetooth capabilities. The RPi 3 also has the greatest computation power, but costs the most out of the RPis. The RPI Zero W is not as powerful, but requires the least power to run and has the lowest cost by far. For the above reasons and the university's availability of the Raspberry Pi in this project, it would be sensible to use either an RPi 3 or RPi Zero W.

2.5 Power

The comparison of SBCs in Section 2.4.4 concluded that either the Raspberry Pi 3 or Zero W would be the most suitable SBC to base the sensing device on for this project. Consequently, this section focuses on powering a Raspberry Pi.

2.5.1 Raspberry Pi power requirements

A Raspberry Pi requires a 5V power source that connects via micro-USB. Table 2.3 shows the current requirements for different Raspberry Pi models. The RPi Zero W has the lowest power needs of the compared models with a minimum required current of 150mA. The RPi 3 requires 400mA at a minimum. However, these are optimistic figures as it is likely that extra current will be needed to also provide power to the sensing board, which will hold the air quality sensors. Considering the aforementioned values, the absolute minimum power needed would be 0.75W (0.15A $\times$ 5V).

2.5.2 Power sources

For vehicle-based devices, an implicit requirement is that the power source must be portable. There are two main options that appear viable to satisfy such a requirement: (i) battery-based, and (ii) energy harvesting.

A battery pack could provide the power for a Raspberry Pi as long as the voltage was 5V. Four AA batteries could provide 6V (1.5V each) and then a voltage regulator could be used to reduce this down to 5V to meet the Pi's requirement. This is a simple solution that would be reliable and provide consistent power.

The second option is to use some form of energy harvesting from the vehicle the device is attached to. These techniques would be useful for making a self-sustaining monitoring system. One example is to use the weaving motion of a bicycle whilst the user is cycling. Yang, Yeo and Priya (2012) studied this form of energy harvesting, but found that it only produced 6.6mW, which is far too low for powering a Raspberry Pi. Another study by Hui (2011) found that using a tyre-driven dynamo could generate power in the range 7-15W. This would be easily sufficient to power a RPi and is a potentially viable option. Other energy harvesting techniques include using the vibrations to generate piezo-electricity or

generating energy via solar panels. However, even if it is possible to generate enough current to run the RPi, this current needs to be produced constantly in order to run the device. Vehicles are prone to stopping in traffic and at junctions, which would lead to the generating current cutting out. To solve this problem, a capacitor would need to be included in the circuit for storing energy. There are also cost implications of including a dynamo. Given batteries are a low-cost and readily available power source for the RPi that can provide consistent power, they seem the more sensible option.

Furthermore, it is critical that the power source can provide energy for a suitable amount of time so that a sufficient number of measurements can be collected. Again, energy harvesting compares poorly to a simple battery solution as it needs to harvest and store the energy as the user cycles. Standard AA batteries have a capacity of between 600 and 2850mAh[1], which is enough to power a RPi Zero W with a current draw of 150mA for between 4 and 19 hours.

For the above reasons, it is apparent that energy harvesting is not practical as a power source in this project. Therefore, it is not considered further.

2.6 Use of low-cost sensors

The recent emergence of low-cost sensors as alternatives to high-cost sensors such as the DustTrak 8530, which costs around £8,000, has enabled low-cost and widespread monitoring of air quality that was previously only carried out by professional personnel. Historically, professional training has been required to operate the equipment and regular maintenance is needed in order to ensure high data quality and accuracy. These tasks can be avoided by using the low-cost sensors.

However, a major problem is that the data generated by these low-cost sensors is often of questionable accuracy and reliability. The causes of this are varied (Clements et al., 2017). Firstly, low-cost sensors are often sensitive to environmental conditions such as humidity and temperature. Secondly, the sensor readings are known to change as they age, with sensor drift leading to the frequent need to recalibrate, thus reducing their cost advantage. Furthermore, they are sensitive

[1]Source: `https://en.wikipedia.org/wiki/AA_battery`

to the air flow, which is particularly poignant given the prevalence of mobile air quality monitoring deployments. Finally, the low-cost sensors can be exhibit a delayed response to a change in the pollution level and a sensor for one pollutant can exhibit cross-sensitivity to a variety of other pollutants. As a result of these issues with low-cost sensors, calibration techniques have been developed to help improve the data quality.

2.6.1 Calibration of sensors

Calibration of sensors is key to ensuring that accurate and reliable measurements are obtained. This topic has particular gravitas for low-cost sensors because of their lower accuracy than more expensive alternatives.

In 2008, the European Parliament (2008) issued a directive on ambient air quality and cleaner air in Europe, in which it laid out a set of Data Quality Objectives (DQOs) that specify the maximum level of acceptable uncertainty in air quality measurements. These are detailed in Table 2.4. If low-cost sensors are to be a viable for meaningful air quality monitoring, their measurements will need to meet or surpass these DQOs.

Table 2.4: EU Data Quality Objectives

Gas/particulate	Maximum uncertainty
CO	25%
NO_2	25%
SO_2	25%
$PM_{2.5}$	50%
PM_{10}	50%
O_3	30%

One simple calibration technique is to calibrate each sensor to a 'ground truth' measurement from a more expensive sensor. One study calibrated low-cost particulate sensors using a device 400 times more expensive and found the sensor provided an accuracy of over 90% (Thorpe, 2017). Typically, linear regression has been used for sensor signal normalisation to reference measurements. However, there is no evidence that these correlations are transferable to different locations (Clements et al., 2017). Furthermore, the study by Thorpe (2017) was conducted

indoors and in a controlled environment, which does not involve the same difficulties as outdoor air quality monitoring using low-cost sensors. Indeed, linear calibration models developed in a laboratory environment have been shown to perform poorly on ambient data (Castell et al., 2017).

To counter these problems, multi-variate linear regression models have been used (Spinelle et al., 2015, 2017), which provide an improvement to the calibration because they take into account more complex factors that are known to affect sensor performance. They also tested artificial neural network (ANN) calibration. ANNs are a supervised machine learning technique that trains the model on known input-output pairs and then uses the trained model to predict the output from unseen inputs. The authors found in both papers that multi-variate linear regression had the highest measurement uncertainty along with the simple linear regression model when compared to an artificial neural network (ANN) for calibration of NO, NO_2, CO, CO_2 and O_3 sensors. Despite the high uncertainty, it was still possible to exceed the DQO for O_3 using a linear regression model. In addition, the ANN approach resulted in exceeding the DQOs for O_3 and CO, and generated an extremely low uncertainty level for CO_2 even though this gas does not have a DQO. On the other hand, meeting the NO and NO_2 DQOs remained a challenge, which may have been as a result of their high cross-sensitivity. Therefore, despite this supervised machine learning technique providing more effective calibration than linear or multi-variate linear regression, further work is needed to calibrate sensors so they comply with the DQOs.

Another machine learning technique uses random forest-based machine learning algorithm. Zimmerman et al. (2018) were the first to apply this technique to low-cost air quality sensor calibrations. Again, it was found that a machine learning technique (random forests) outperformed multivariate linear regression models and was able to make the low-cost sensors accurately characterise air pollution concentrations synonymous with European city levels. Moreover, by including measurements of additional pollutants, the random forest model was able to account for pollutant cross-sensitivities, which Spinelle et al. (2015, 2017) struggled with. This highlights the need of having multiple sensors in the same device.

Whilst it appears machine learning methods are the best available methods for low-cost sensor calibration, the aforementioned studies were used stationary air

quality monitoring. Thus, it is unclear how these methods would perform when calibrating in near real-time. This should be investigated further.

2.6.2 Air flow

Since low-cost sensors are sensitive to the air flow, it is important that the sensors are provided with a consistent air flow so as to not affect their measurements.

One study has noted the benefit of air flow in improving air quality measurements from low-cost sensors. Thorpe (2017) found that the increased air flow from using an integrated fan improved the signal to noise ratio of a particulate sensor. The fan provided a more consistent air flow to the sensor, which may have been the reason for the improvement. However, the author noted that battery draining may have led to a variable fan speed, which in turn could have affected the measurements. Thus, variability in air flow would need to be controlled for in any modelling.

In addition, Hasenfratz et al. (2015) used a fan installed in the back of their air quality monitoring 'box' to draw air out and ensure a steady air flow.

An alternative to using a fan for improving air flow is to funnel the air flow directly onto the sensor. This could be achieved through the use of a cone-shaped device. Unfortunately, a literature search did not discover any research relating to this application. Therefore, this should be researched further.

2.7 Mapping air pollution

Given the health impact of air pollution detailed in Section 2.1.1, maps are needed to identify pollution 'hot spots', to show changes in pollution over time, to define at-risk groups and to provide better air pollution exposure estimates for epidemiological studies.

2.7.1 Estimation techniques

As discussed in Section 2.3.3, current air quality monitoring mainly uses fixed monitoring stations. Even in the 1990s, it was noted that air pollution often varies over extremely short distances whilst the air quality data from these fixed

stations are often sparse, leading to highly generalised maps and poor estimates (Briggs et al., 1997).

As a result, techniques have been developed to generate estimated air pollution values in areas where no readings have been taken, thus allowing the improvement of air pollution maps by increasing the resolution. Such methods include spatial interpolation, dispersion modelling and regression-based approaches. A specific example of a regression-based approach is land-use regression (LUR), which was first detailed by Briggs et al. (1997) using data from the SAVIAH (Small Area Variations in Air Quality and Health) study. It has become increasingly popular since its introduction (Hoek et al., 2008). This method combines air quality measurements spread over an area with stochastic modelling in order to generate air quality predictions at unsampled locations. The independent variables for the regression are obtained through geographical-information systems (GIS), which contains data on various environmental factors such as traffic volume and the road network. Figure 2-3 visually demonstrates this method. y is the pollution concentration and x_i are the land-use types within the 'buffer' (circles). Good performance of the regression maps was achieved, with r^2 values ranging from 0.79 to 0.87 across the three mapped locations. However, LUR models are poor at separating out the impact of different pollutants due to the high correlation that can occur between pollutants (Hoek et al., 2008) and this type of extrapolation can perform poorly when the study area has significantly different land-use. For example, moving the study area from Bath, which is a small city in the countryside, to London, which is a much bigger and congested city.

GIS files are made publicly available from DEFRA for modelling purposes for local air quality management. The relevant file could be incorporated with the data obtained in this project to generate pollution maps of the local area that includes unsampled locations.

2.7.2 Mapping visualisations

A particularly interesting visualisation of geo-located air quality data is a heatmap. Google's API has a built-in Heatmap Layer for client-side rendering of heatmaps, which can be used to visualise data in this way. Figure 2-4 shows an example from their website.

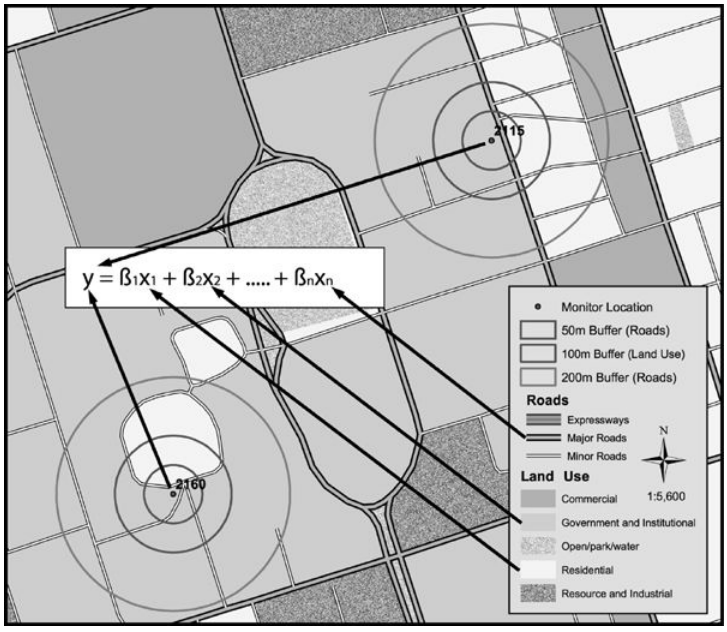


Figure 2-3: Visual representation of LUR by Jerrett et al. (2005)

Figure 2-4: Heatmap Layer example from the Google API website (Google, 2017b)

Devarakonda et al. (2013) successfully used this Heatmap Layer for real-time air quality monitoring. Their air quality data was stored in a Google Fusion Table, which is part of their cloud offering and acts like a database, providing seamless integration between the data and Google's visualisation tools. The air quality data is shown neatly on top of the map with the red/green colour scale representing high/low pollution respectively. From this map, hotspots can quickly be linked to specific road layout points. For example, the authors identified a hotspot at an

intersection where traffic was merging. This indicates that this method is suitable for identifying troublesome areas of high pollution.

An alternative is to use a custom coloured shape overlay on a map. For example, Hoang et al. (2013) used hexagons, shown in Figure 2-5. The hexagonal shapes were used instead of the more widely used rectangular shapes because of their visual advantages. Firstly, there is less map ambiguity, and secondly, hexagonal shapes are easier for human vision to interpret since, on the map, there are no distinct horizontal and vertical lines, to which the authors note human vision is especially sensitive to.

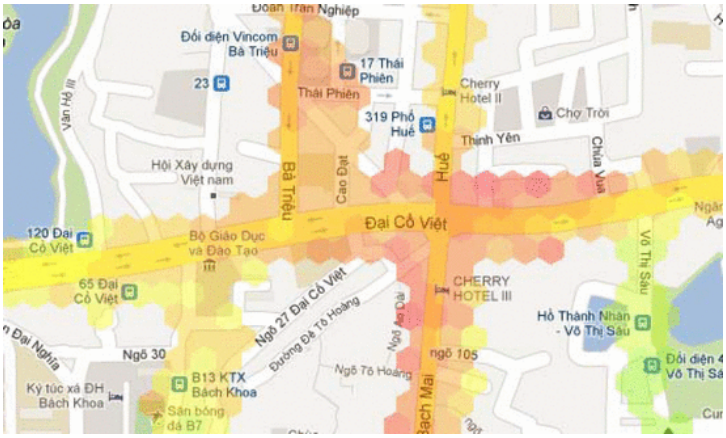


Figure 2-5: Heatmap using hexagonal shapes (Hoang et al., 2013)

2.8 Health-optimal route planning

One potential application of air quality data is in alternative route planning for the public that takes into account pollution levels on different streets. Typically, route planning has focused on finding the shortest route from an origin to a destination. However, recent developments have begun to incorporate other factors. For example, [INSERT REFS FOR DIFFERENT ROUTING]. In addition, air pollution has begun to be incorporated into route planning (Sharker and Karimi, 2014; Hasenfratz et al., 2015).

Hasenfratz et al. (2015) use trams in Zurich to generate high-resolution pollution maps, which are then used to identify pollution levels on every street in

a given section of the city. They use the product of the pollution level and the length of the road segment to add weights to the roads. These weights are then used by the A* search algorithm to generate lowest air pollution exposure routes. The authors find that travelling along these alternative routes leads to a 7.1% reduction in air pollution exposure for a 6.4% increase in route length.

Sharker and Karimi (2014) conducted a similar US-based study. The authors overcame the issue of low spatial resolution data by assigning weights to roads from sparse fixed monitoring stations using two different interpolation methods: (i) Inverse Distance Weighting (IDW), and (ii) Kriging. The authors used Dijkstra's algorithm for their search algorithm. Overall, they found an 11.8% reduction in pollution exposure at a cost of an 8.6% increase in route distance by using the alternative routes. They also looked at temporal changes in road weightings and found them to be higher during the daytime, especially around midday. Interestingly, they found travel time to be lower on average for lower exposure routes. This was attributed to travelling more along highways, which have a higher speed limit, in order to avoid congested road segments. Whilst the author did not focus on a specific vehicle type, this finding may not relate well to cyclists because the larger and faster roads are often more dangerous, which may deter cyclists from cycling along them. Furthermore, cyclists cannot cycle along motorways in the UK.

2.9 Data quality monitoring

Outlier/anomaly detection is important in the process of data cleaning. An outlier is defined as a value that is statistically different from other similar observations. Since the low-cost sensors are known to have accuracy issues (see Section 2.6), it is particularly relevant to data from low-cost sensors. To ensure the quality of the data, these will need to be identified and then corrected or removed from the data.

A simple method of identifying outliers is checking whether an observation falls within a certain confidence internal or not. A recent application of this to air quality monitoring comes from van Zoest, Stein and Hoek (2018). They first divided up NO_2 data into spatio-temporal classes because different classes may have different characteristics that affect whether a measurement is an outlier

or not. For each class, a threshold was calculated, from which outliers could be detected. However, care must be taken to ensure that an extremely high pollution measurement is an error and not caused by an event, such as extreme traffic. The authors also noted that future research is needed to apply this method to real-time outlier detection.

Another method of identifying these erroneous measurements is through the use of automatic outlier detection. This term applies to a wide range of techniques, but one such technique is k-nearest neighbours in which the distance between a data point and the kth nearest data point is calculated. The greater this distance, the more likely the observation is an outlier. A different technique is clustering in which data points are grouped into sets of similar data points. Observations deemed not to be in any cluster are classified as outliers.

There appears to be a lack of papers applying outlier detection to air quality monitoring, particularly for more advanced outlier detection methods. This could be a potentially new application that needs to be researched further.

2.10 Summary

Air pollution is a major problem around the world, with serious negative health consequences. As a result, the WHO (2002) have identified a clear need for monitoring air pollution and governments have put great emphasis on identifying which pollutants are most worth targeting. However, to effectively monitor air pollution, they need a low-cost, ubiquitous system of monitoring devices, but current government monitoring systems have a significant cost barrier. Previous work has attempted to provide a cheaper and more portable method of measuring air quality than using industrial-grade sensors, with numerous studies using a Raspberry Pi as the computer upon which the sensing platform is based. These studies have implemented various vehicles, including bicycles, tricyles, cars and trams. This mobility advantage over fixed monitoring systems has led to an improvement in the spatial resolution of the measurements. In addition, the increased availability of low-cost sensors has enabled the cost reduction, although at the cost of lower accuracy. As a consequence, methods such as machine learning have been developed in order to improve their accuracy. This air quality data can then be presented to the public and be used to inform public health decisions,

such as health-optimal transport routing and where to build bike paths.

This literature review has identified a number of gaps in the literature that this project will attempt to address. These are as follows:

1. Creating an air quality monitoring device that measures a number of the pollutants set out by the UK and EU's air quality guidelines
2. Assessing the capabilities of a Raspberry Pi for air quality monitoring purposes
3. Whether using a bicycle is a suitable vehicle for air quality monitoring
4. Whether a number of sensing capabilities can be incorporated into one, sufficiently small device
5. Assessing the air flow needs of bicycle-based sensors
6. Applying outlier detection methods to air quality data
7. Mapping air pollution accurately from low-resolution data by using the techniques detailed in Section 2.7.1
8. Whether air pollution data can be incorporated into route planning for public benefit

Chapter 3

System design and implementation

In order to answer our research questions detailed at the end of Chapter 2, we developed an air quality monitoring device that attaches to a bicycle. The device is based upon a Raspberry Pi computer and includes a variety of different sensors. These are as follows: temperature and humidity sensor, particulate (PM) sensor, ozone (O_3) sensor and a general air quality sensor that is sensitive to a variety of gases that includes nitrogen oxides (NOx) and carbon monoxide (CO). It also has a GPS module attached that provides geo-location information. The device sends the collected data to a database, which is then processed and displayed in a web application.

In this chapter, we describe the design and implementation of the air quality monitoring device, providing our rationale for both the physical construction and the software design.

3.1 Physical device

This section details how the device is constructed. An image of the final device is shown in Figure 3-1.

Figure 3-1: Picture of the finished device. (a) shows the outside view, whereas (b) shows inside the case.

3.1.1 Technical details

The core of the sensing device is a Raspberry Pi Zero W single-board computer, with a 1 GHz CPU running the Raspbian Stretch, Linux-based, operating system[1]. The sensing capabilities come from a Shinyei PPD42 particulate sensor, a MikroElektronika Ozone 2 Click, a MikroElektronika Air Quality Click and a DHT22 temperature and humidity sensor. The MikroElektronika sensors are connected to the RPi using a MikroElektronika Pi 3 Click Shield. A GlobalSat BU-353S4 USB GPS receiver supplies the precise geospatial information for geo-tagging the air quality measurements. The built-in WiFi of the RPi is used to connect to a WiFi hotspot in order to upload the measurements to a MySQL database (version 5.7.16 Community Server) via a PHP script. An Android mobile phone running Android 6.0.1 is used to provide the WiFi hotspot whilst cycling. The program running on the RPi is written in the Python programming language and can be found in Appendix XXX. The power for the device is supplied via a 10000mAh external battery pack, with a 5V 2.4A output. The web application is written in R and uses R Shiny to generate the interface.

[1]Link to download: https://www.raspberrypi.org/downloads/raspbian/

3.1.2 Raspberry Pi

The original plan uses a Raspberry Pi Zero W as the base computer, but in testing an overheating issue was encountered, rendering an RPi completely useless. The device's CPU was running extremely hot, which is potentially a side-effect of the device containing a large number of sensors. RPis do not come with heat-sinks, but these could be included in future designs. To fix the issue of overheating, we switched to a Raspberry Pi 2. The Raspberry Pi 3 was also tested as an alternative, but the Serial Peripheral Interface (SPI) setup is different and so the software did not run correctly (see Section 3.2.1 for SPI details). The device does not need the additional computing power, so the Raspberry Pi 2 is deemed to be satisfactory.

3.1.3 GPIO

GPIO stands for General Purpose Input/Output and provides a standardised interface for the RPi to a variety of other devices. The device's RPi has 40 GPIO pins, of which 17 are used. The active pins are detailed in Table 3.1. Given the number of sensors attached to the device, it is difficult to connect them all to the GPIO pins of the RPi (at a minimum there are power, ground and signal wires). To overcome this, a GPIO expander is used, with care taken to ensure that GPIO pins are only used for a single signal (issues arise if multiple signals are sent on the same pin).

3.1.4 Particulate sensor

A number of different low-cost particulate sensors are considered for this project, based upon previous research. These are detailed in Table 3.2. Garnier (2017) evaluated the Sharp GP2Y and HK-A5 particulate sensors, finding that both of these sensors had low to zero correlation with an industry-grade DustTrak 8533 particulate sensor, despite controlling for external factors. Thorpe (2017) also evaluated these sensors, as well as the Shinyei PPD42 sensor. The HK-A5 sensor was found to have no response to changes in particulate concentration in preliminary testing and was subsequently excluded from the author's investigation. The author, unfortunately, could not provide much evaluation of the Sharp

Table 3.1: GPIO pin layout and usage for the 40 GPIO pins on the Raspberry Pi. Of these 40, 17 are directly used by the sensors in the device.

Pin	BCM number	Usage	Pin	BCM number	Usage
1	3v3 power	DHT22 power	2	5V power	Shinyei power
3	BCM 2		4	5V power	MQ135/131 power
5	BCM 3		6	Ground	Shinyei ground
7	BCM 4		8	BCM 14	
9	Ground	DHT22 ground	10	BCM 15	
11	BCM 17		12	BCM 18	
13	BCM 27		14	Ground	
15	BCM 22	DHT22 signal	16	BCM 23	Shinyei signal
17	3v3 power		18	BCM 24	
19	BCM 10	MQ131 MOSI	20	Ground	LED ground
21	BCM 9	MQ131 MISO	22	BCM 25	LED signal
23	BCM 11	MQ131 SCLK	24	BCM 8	MQ131 signal
25	Ground		26	BCM 7	
27	BCM 0		28	BCM 1	
29	BCM 5		30	Ground	
31	BCM 6		32	BCM 12	
33	BCM 13		34	Ground	
35	BCM 19	MQ135 MISO	36	BCM 16	MQ135 signal
37	BCM 26		38	BCM 20	MQ135 MOSI
39	Ground		40	BCM 21	MQ135 SCLK

Table 3.2: Comparison of different particulate sensors considered. All of these sensors use a light-scattering technique for detecting particles. The Shinyei PPD42 sensor is the cheapest, with joint-smallest minimum particle detection size.

Sensor	Sensor type	Minimum PM size	Cost (£)
Shinyei PPD42NS	Light scattering	PM_1	7.75
Sharp GP2Y	Light scattering	PM_{10}	9.46
HK-A5	Light scattering	PM_1	37.70

GP2Y sensor due to software bugs and damaged sensors. As a result, Thorpe focused on the Shinyei PPD42 sensor and found it to track changes in particulate concentration reliably, with a substantial accuracy improvement through the use of polynomial fitting or a single feature multi-layer perceptron. Therefore, the Shinyei particulate sensor is a sensible choice for use in this project.

The chosen Shinyei PPD42NS sensor has two output lines, named P1 and P2.

P1 is used to measure particles between $1\mu m$ and $10\mu m$, whereas the P2 output is used to measure particles between $2.5\mu m$ and $10\mu m$. In testing, we rarely registered a signal from P2 that differed from P1, which would indicate that the sensor only detects particles with diameter above $2.5\mu m$. This implies the output from the sensor is classified as PM10. This agrees with the findings of Thorpe (2017), who found that the sensor responded very poorly to smoke which has an average particle size of between 1 and $2\mu m$.

The sensitivity of this sensor is a clear drawback if accurate measurements of PM2.5 are needed. However, we are more interested in creating a working air quality monitoring device and testing the effectiveness of a bicycle-based device, meaning a lower accuracy can be afforded. Moreover, PM10 is still a governmentally monitored pollutant in the UK (see Table 2.1) and provides a solid measure of air quality. Nonetheless, it is still worth noting this issue for future developments.

3.1.5 Ozone sensor

For the gas sensors, we opt for using MikroElektronika's click-board technology, which provides a homogeneous platform that a variety of sensors can be plugged directly into without the need for soldering. This dictates our choice of ozone sensor because MikroElektronika only have one option available – the 'Ozone 2 Click'. This click-board uses an MQ131 sensor on the circuit board, which has a tin dioxide sensing layer that reduces in conductivity in clean air. It can detect ozone concentrations between 10 and 1000 parts-per-billion (ppb).

One of the main reasons for choosing to use MikroElektronika's click-board technology is the modular nature of the sensors, which all use the same 16-pin design. This means we can easily change sensors quickly to suit our requirements and re-purpose the air quality monitoring device at short notice, therefore avoiding soldering of circuit boards.

The MQ131 ozone sensor has been used in air quality projects previously, including in a unmanned aerial vehicle (UAV) air quality monitoring device (Alvear et al., 2017) and indoor monitoring (Abraham and Li, 2014).

3.1.6 NOx sensor

For an NOx sensor, we chose the MikroElektronika 'Air Quality Click'. This uses an MQ135 sensor and is marketed as being sensitive to NOx. However, upon further analysis, one can see that NOx is not listed on the data sheet's calibration curves and the sensor is actually sensitive to a variety of different gases. Unfortunately, the company did not have any dedicated NOx sensors available to replace this with. As a result of this issue and the project's time constraints, we use it as a general air quality sensor.

Nonetheless, as previously explained, the modular nature of MikroElektronika's click-board technology means it would be easily replaceable in the future. In fact, towards the end of the project, MikroElektronika released a dedicated NO_2 sensor. This is worth noting for future developments of this air quality monitoring device.

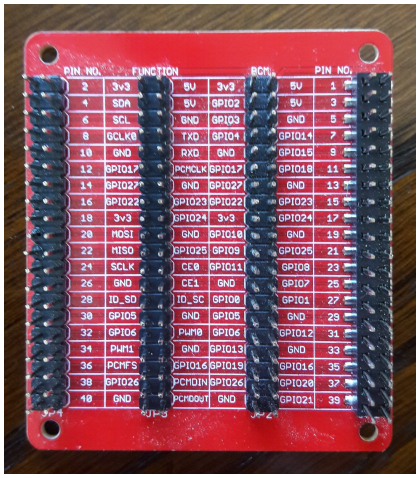
A number of previous studies use the MQ135 sensor for air quality monitoring purposes. Rahman et al. (2017) used the MQ135 in an Internet-of-Things (IoT) based platform, whilst Andersen et al. (2012) used it on a bicycle-based air quality monitoring device. In addition, Garnier (2017) tested the MQ135 and found it to be accurate for monitoring changes and trends in air quality.

3.1.7 Temperature and humidity sensor

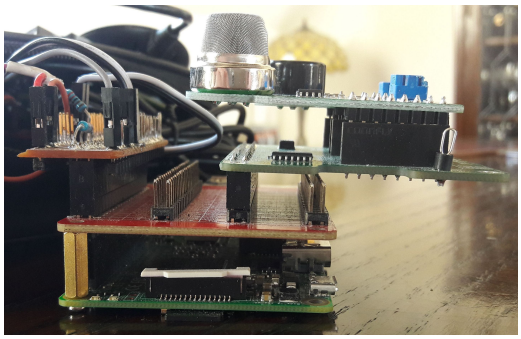
A DHT22 temperature and humidity sensor adds climate monitoring capabilities to our device. Garnier (2017) found the DHT22 to be more accurate than a BME280 sensor (another commonly used temperature and humidity sensor), despite costing less. This sensor has been used frequently elsewhere in the literature (Rahman et al., 2017; Ibrahim et al., 2015; Alkandari and Moein, 2018). By including this sensor in the device, we are able to correct the particulate and gas sensor measurements, which are sensitive to temperature and humidity at certain levels.

3.1.8 HAT

The Mikro Pi 3 Click Shield, which holds the ozone and general air quality sensors, uses a 40-pin GPIO connector to interface with the RPi, although not all of the



(a) GPIO expansion board.



(b) Two HATs connected to GPIO expansion board.

Figure 3-2: Solution to sensor attachment problem. (a) The GPIO expansion board provides 4 rows of 40-pin GPIO connections. (b) 2 HATs are connected to the RPi via the expansion board, which enables our device to have increased sensing capabilities.

pins are used (see Table 3.1). Therefore, in order to attach the particulate sensor, the temperature and humidity sensor and the LED to one RPi, a solution is needed that enables other connections to the GPIO pins.

The chosen solution is a GPIO expansion board (see Figure 3-2a), which provides 4 additional 40-pin GPIO connectors. Figure 3-2b shows how the multiple connections are made. The particulate sensor has a digital output and so can be connected directly to the GPIO pins with the use of a voltage divider made out of a $2.2k\Omega$ and $3.3k\Omega$ resistor. The temperature and humidity sensor and LED also have digital outputs, meaning they are connected directly to the GPIO pins as well. These connections to the GPIO expansion board are made more robust through the use a custom-built circuit board. A protoboard has a GPIO connector soldered to its underside, which enables it to connect to the GPIO expansion board. Single lines of GPIO pins are also soldered on, along with the relevant resistors discussed previously. Female-to-female jumper cables connect the sensors and the LED to these GPIO pins. Figure 3-3 shows the final soldered protoboard.

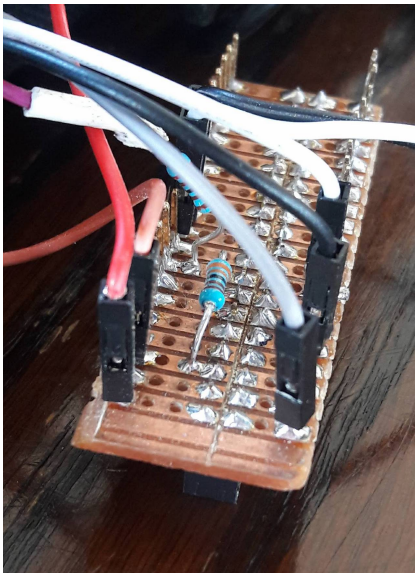
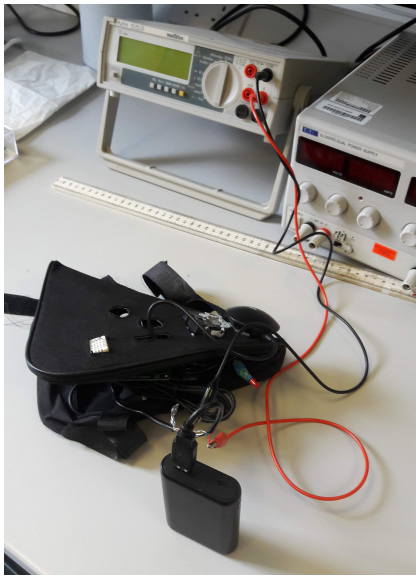


Figure 3-3: Soldered protoboard. This custom circuit board enables the connection of the sensors with digital outputs (Shinyei PPD42 and DHT22, as well as the LED) to the RPi.

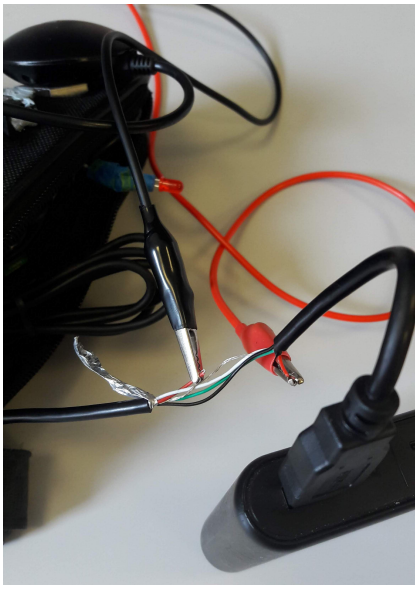
3.1.9 Power

The power to the device is supplied by a 10000mAh lithium-ion battery. The current draw of the whole device can be measured using an ammeter, dissecting a USB cable and attaching clips to the voltage common collector (VCC) wire, commonly known as the 'power' wire. This test setup is shown in Figure 3-4a, with the dissected cable shown in Figure 3-4b. The ammeter reported current values between 748mA and 754mA. Therefore, the device can run for approximately 13 hours on a fully-charged battery. It must be noted that Equation 3.1 is only an estimate because batteries typically have non-linear discharge due to factors such as temperature and age of battery.

$$t = \frac{10000mAh}{750mA} = 13.\dot{3}\ hours \tag{3.1}$$



(a) Connection to ammeter.



(b) Dissected USB cable with VCC wire exposed.

Figure 3-4: Power consumption testing. (a) shows the test setup with a connection to an ammeter, and (b) shows how a USB cable is dissected to enable sampling of the current drawn.

Table 3.3: Cost breakdown of device. The overall cost of the device is relatively low at only £152.63, which satisfies this project's objective of making a low-cost air quality monitoring device.

Item	Cost (£)
Raspberry Pi 2 Model X	20.00
ModMyPi Triple GPIO Expansion Board	7.99
Shinyei PPD42NS particulate sensor	7.75
DHT22 temperature and humidity sensor	4.99
Mikro Ozone 2 Click	33.90
Mikro Air Quality Click	15.91
Mikro Pi 3 Click Shield	15.60
GlobalSat BU-353S3 USB GPS receiver	30.99
10000mAh external battery pack	15.00
Jumper cables	0.50
Total	152.63

3.1.10 Case

The case is a modified version of a triangular bag that fits to the frame of a bicycle. It has three velcro straps that secure the bag to the bicycle's frame. The inside of the bag is lined with a flexible plastic, thus helping to make the device water-resistant. There are holes cut into the side of the bag to: (a) expose the air quality sensors to the air, (b) to allow the GPS module to have a clear view of satellites in the sky, and (c) to display a system-status LED to the user.

3.1.11 Cost

One of the main aims of this project is to create a low-cost device. In total, the cost of a complete air quality monitoring device is £152.63. A breakdown is shown in Table 3.3.

Given the device uses the mobile phone's cellular network to upload the measurements, it is also worth noting the data plan cost. The device generates 542 bytes/minute of data on average when sampling every 10 seconds. With an average running time each day of 2 hours, the device will use around 2 MB of data per month. Assuming a 5GB data plan per month costs £10, the data consumption of this device costs £0.004 per month. This is negligible.

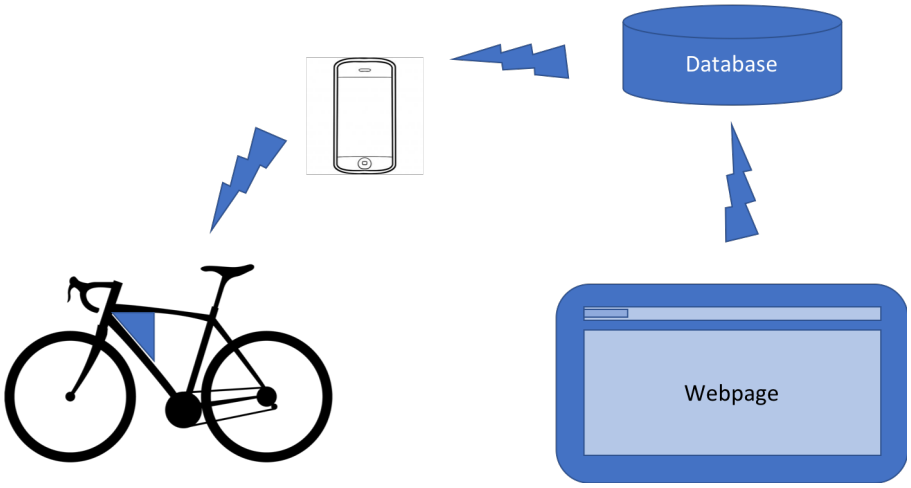


Figure 3-5: Data flow. The device attached to the bicycle sends data to a database using the phone's internet connection, which is then displayed on a webpage.

3.2 Software

The software for this air quality monitoring device covers a number of different areas.

1. Software that runs on the Raspberry Pi to collect and send the sensor data (Section 3.2.1)
2. Server-side software that receives the data and inserts it into a database (Section 3.2.2)
3. Web application that displays the data to the public (Section 3.2.3)

3.2.1 Raspberry Pi software

The main requirement of the software on the Raspberry Pi is to read the data from the sensors periodically and then send this data to a specified server. The basis for this software came from analysing code from the ENLITEN project [REF], a previous study that the University of Bath was involved in, as well as from Thorpe (2017).

Although Thorpe (2017) used the same particulate sensor, the study used a Grove Pi HAT, which was based around an Arduino and the software only allowed for measurements every 30 seconds. We worked with the manufacturer Grove to reduce this, but encountered numerous problems with re-flashing the Grove Pi with altered firmware. To overcome this, we instead attach the sensor to the GPIO pins and use the `pigpio` python module, which provides control of the RPi's GPIO. Furthermore, we use the `spidev` library for communicating with the MQ131 and MQ135 sensors (more specifically, their ADC chips). `spidev` is a python module for interfacing with SPI devices via the `spidev` linux kernel driver. A problem can arise when attempting to read from all of the sensors simultaneously, because the continual nature of reading from the particulate sensor (measuring how long the signal was low – i.e. 0 – for in a given time period) 'blocks' the main thread. As a resolution, the software implements two threads: (i) reading the particulate sensor and operating the LED, and (ii) operating the rest of the sensing functionality. This avoids any blocking and the sensor measurements are saved to variables in the relevant thread objects, which can then simply be accessed from the object. Every 10 seconds, these variables are read by the main program and saved to a CSV file. Once the CSV file has 6 lines of data, corresponding to 1 minute of time, the software creates a new file.

To avoid manual intervention by the user, the `/etc/rc.local` ('rc' stands for 'run-control') is edited to include a start service command, shown below. This automatically runs the program when the Raspberry Pi boots. An alternative, `cron`, is available to run scheduled jobs, but was found to be more difficult to get working on a reboot trigger.

```
sleep 30 && sudo -H -u pi python /home/pi/filename.py &
```

To provide an auto-connect feature for the WiFi, the Android phone's WiFi hotspot details are saved in the Raspberry Pi's `wpa-supplicant.conf` file, which means it automatically connects when it detects the WiFi network. An example config file is shown in Appendix XXX. The `requests` python module is then used to send the CSV file via HTTP post requests over the RPi's WiFi connection.

These two requirements satisfiy the basic functionality of the RPi software. However, for robustness, the software stores un-sent data on the RPi until it can

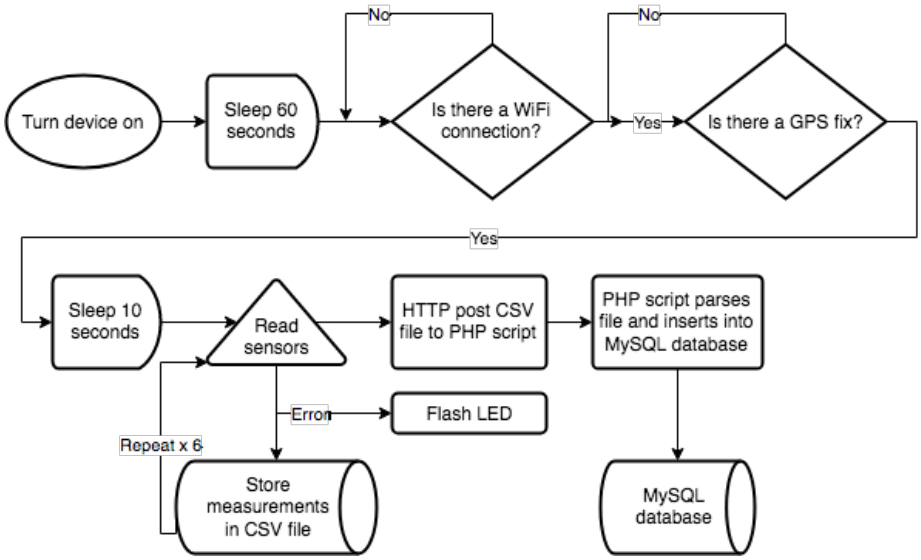


Figure 3-6: Flow diagram of system code.

be sent again. For example, if the WiFi connection fails, then the files are stored in a separate directory and then iteratively sent when the connection is restored. In addition, the software detects potentially erroneous measurements (e.g. a zero reading) and alerts the user by flashing a red LED. This functionality was achieved through the use of the `RPi-GPIO` library.

Figure 3-6 visualises this whole process, as well as the insertion into the database.

GPS

Geo-location functionality is provided by a USB GPS module, specifically a GlobalSat BU-353S3 USB GPS Receiver. This plugs directly into one of the RPi's USB ports and appears as the device `ttyUSB0`. The GPSD protocol, which is built on top of JSON, is used to communicate with the GPS receiver. Responses from the GPS receiver are JavaScript Object Notation (JSON) objects. The `gpsd-py3` python module is used to interface with the GPSD protocol, parsing the JSON to output longitude and latitude coordinates, as well as other useful information such as horizontal speed and altitude.

Moreover, the RPi is configured to use the GPS receiver as a Network Time

Protocol (NTP) time server. This involves the use of the NTP daemon. Installation instructions are detailed in Appendix XXX. By enabling this, the RPi syncs its internal clock with an accurate source – the GPS satellites. This overcomes the issue of the Raspberry Pi having no real time clock, which otherwise can lead to inaccurate timing when recording measurements because the RPi is oblivious to time lapsed since the previous shutdown.

Initially, the project was going to develop an Android application that merged readings from the RPi with GPS data, because the RPi does not have a built-in GPS receiver. The application was created using Android Studio for Mac OSX. The application works by connecting the RPi to the Android phone via Bluetooth, and then uses `ObexFTP` to send the CSV files from the RPi to the phone. The application then watches a particular directory for new files and upon receipt matches up timestamps in the CSV file with GPS-timestamp pairs stored on the phone, before HTTP posting the file to the server. However, this was a far from ideal solution and introduced tight coupling between the RPi and the phone. Once it was realised that a USB GPS module was available, the Android application was shelved.

Although not used in the final version of the device, this development shows that it is possible for the same functionality to be achieved using Bluetooth. It also demonstrates the ability to still collect GPS location data when the Raspberry Pi does not have GPS itself and no GPS module is available. Since the cost of the GPS module is £30.99, this alternative solution can be used in an even lower cost version of the device.

3.2.2 Server-side software and database

In order to connect the air quality monitoring device and the database, a client-server architecture is used. A RESTful API receives the CSV files, parses them and then inputs the file's data into a database.

RESTful API

A RESTful API written in PHP receives the HTTP posted CSV file, then parses the contents and inserts each line into a row of the database table that stores all measurements. If this completes with no errors, it then also inserts a new row

into a system monitoring table with information about the CSV file and time of insertion. If there is an error, a flag is raised in the table that can then be reviewed to ascertain the cause. Appendix B.2 shows the API code.

Database

The University of Bath provides a database hosting service to members of the university. The database software is MySQL version 5.7.16 Community Server. One of these databases is used in this project, containing two main tables: (i) a table that stores the air quality data, and (ii) a table that stores system monitoring information. We are able to only use one table for storing the measurements as the number of rows is low enough to not cause performance issues. The system monitoring table allows identification if and when an error occurs in the data flow.

3.2.3 Web application

The web application is written in R and uses R Shiny version 1.1.0 to provide the web framework. An example page from the web application is shown in Figure 3-7.

The web page's main requirement is to display the information in an easily accessible and understandable manner, so that any member of the public is able to analyse the pollution in their area. It predominantly displays the pollution maps of Bath and line graphs for the time series of data. On the left-hand side of the page is a menu bar that allows the user to select their chosen time period, with the graphs dynamically updating as the selection changes. In addition, there is a dynamic table that displays all of the raw data, with a button for downloading the dataset as a CSV file.

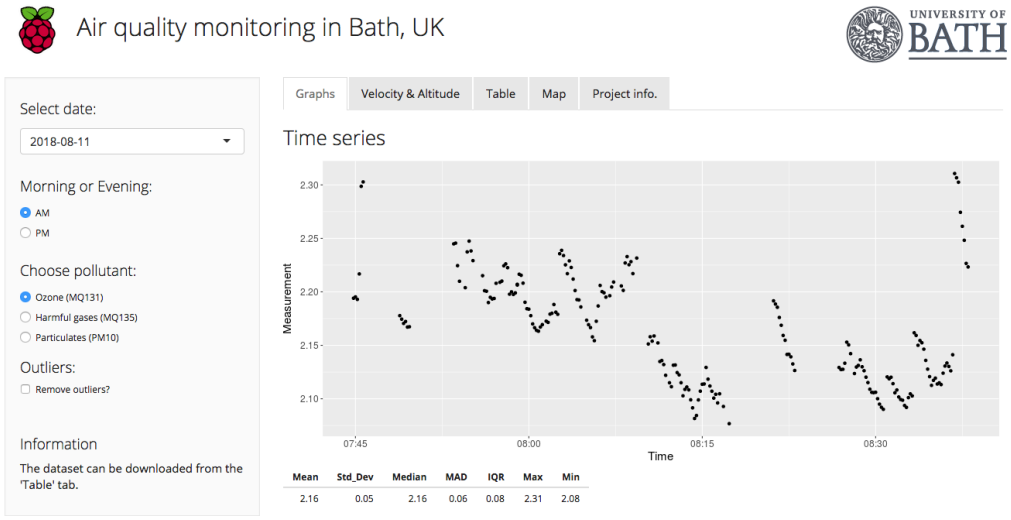


Figure 3-7: Web application using R Shiny. The menu on the left-hand side allows the user to select the time period and pollutant, as well as whether outliers should be included in the graphs or not. There are 4 tabs to the webpage, which each display a variety of graphs and tables to the user.

Chapter 4

Output conversion and calibration

The previous chapter detailed how the device was constructed and provided a high-level description of the software design, although it did not focus on how the sensor outputs were read and interpreted at the low-level.

Sensors either output an analog or digital signal. If the signal is analog, an analog-to-digital converter (ADC) needs to be used to make the output binary (low or high voltage). The communication protocol then must be studied in order to understand how the digital signal can be read in order to get a meaningful signal, instead of reading randomly on the relevant GPIO pin. Once the binary value has been recorded, it then needs to be converted to a decimal value.

The second stage is then interpreting this decimal value and working out how it relates to a meaningful reading. For example, we may read a value of 2048 from a sensor, which could correspond to 2.5V, but the voltage then needs to be mapped to a concentration of a gas.

This chapter details how these processes were created for each of the sensors used in this project and is relatively low-level in comparison to the other chapters.

4.1 Particulate sensor

The output of the Shinyei PPD42NS particulate sensor is a digital pulse. To infer the measured concentration, a value called the Low Pulse Occupancy (LPO) time

is calculated. This is the percentage of time over a fixed period that the digital output of the sensor is 'low' (i.e. 0). The LPO can then be used to calculate the estimated number of particles per 0.01 cubic feet using the following equation, which is obtained by digitising the data sheet calibration curve:

$$y = 1.1x^3 - 3.8x^2 + 520x + 0.62 \tag{4.1}$$

However, the DustTrak reference sensor outputs measurements in $\mu g/m^3$, which is an SI unit, meaning the y values from equation 4.1 need to be converted to this unit. An algorithm was developed by Uva et al. (2009) that achieves this by estimating average particle size and density. It must be stressed that this conversion is only an approximation, because it is impossible to precisely infer the properties of each of the thousands of microscopic particles being counted. The main assumptions are that all particles are spherical with average radius r and density d:

$$r = 0.44 \times 10^{-6} m \tag{4.2}$$

$$d = 1.65 \times 10^{12} \mu g/m^3 \tag{4.3}$$

From these assumptions, the volume v of a sphere (particle) can be calculated approximately:

$$v = \frac{4}{3}\pi r^3 = 3.568 \times 10^{-19} m^3 \tag{4.4}$$

The mass m of a particle is its density multiplied by its volume:

$$m = d \times v = 0.589 \times 10^{-6} \mu g \tag{4.5}$$

Therefore, the mass of particles per cubic foot is calculated by multiplying the mass of a particle m by the number of particles per cubic foot y. It is then a simple conversion from cubic feet to cubic metres by multiplying by a factor of 3531.5.

$$\mu g/m^3 = y \cdot m \cdot 3531.5 \tag{4.6}$$

Additionally, there is a humidity and temperature adjustment, described in Equation 4.7. O is the output of the initial conversion from particle count to concentration (described above), H is the relative humidity percentage, C is the

correction factor, and F is the final adjusted output concentration.

$$F = O \times H \times C \tag{4.7}$$

Table 4.1 shows the values of H and C in normal conditions.

Table 4.1: Humidity and temperature factors

H	C
0-39%	13
40-49%	8
50-59%	6
60-69%	4
70-79%	1.75
80-89%	1.5
90-100%	1

4.2 Ozone sensor

The output of the MQ131 ozone sensor is a raw analog signal that represents voltage, with the voltage proportional to the quantity of the gas being sensed. Given that the RPi cannot read an analog signal (digital only), an ADC chip is included on the MikroElektronika sensor board. The ADC is an MCP3551, which is a single-channel 22-bit ADC chip. The communication between the RPi and this ADC is via a serial communication. More specifically, a Serial Peripheral Interface (SPI). This is a synchronous serial communication interface consisting of an SPI master and SPI slave relationship, which is visualised in Figure 4-1.

Each data word is made up of 24 bits (3 bytes): 22 bits of conversion data and 2 additional overflow bits. To read from the MCP3551, the RPi must detect when the chip is ready, indicated with $\overline{CS}$ (Chip Select) low on the $SDO/\overline{RDY}$ (Data Output) pin. Conversely, a high state ($\overline{CS}$ held high) indicates the device is busy converting. When reading from the ADC, the data is passed Most Significant Bit (MSB) first – this is the bit in the binary number with the highest numerical value. If these 24 bits are indexed as bit 23 to bit 0, then bit 23 and bit 22 are the overflow bits, bit 21 is the sign bit (indicates positive or negative), and bits 20

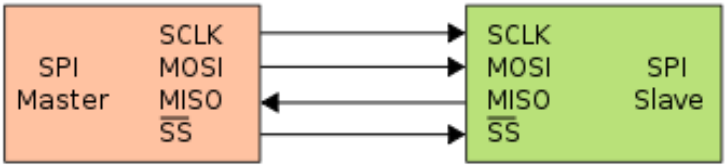


Figure 4-1: SPI master-slave design. This visually depicts the relationship between a single master and a single slave on a Serial Peripheral Interface (SPI) bus. (SCLK = Serial Clock, MOSI = Master Output Slave Input, MISO = Master Input Slave Output, SS = Slave Select.)

to 0 contain the 21-bit value. Excluding the overflow bits, values therefore range from $-2{,}097{,}152$ (2^{21}) to 2,097,151 This information is visualised below in Figure 4-2.

Analog Input Voltage	OVL	OVH	Digital Output Code																				Decimal Code	Hexa
	B23	B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0		*B[23:0]*
Vref +1 LSB	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2097153	600001
Vref	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2097152	600000
Vref - 1 LSB	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2097151	1FFFFF
2 LSB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	000002
1 LSB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	000001
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	000000
-1 LSB	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	3FFFFF
-2 LSB	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	-2	3FFFFE
-Vref	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-2097152	200000
-Vref -1LSB	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-2097153	9FFFFF
-Vref -2LSB	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	-2097154	9FFFFE

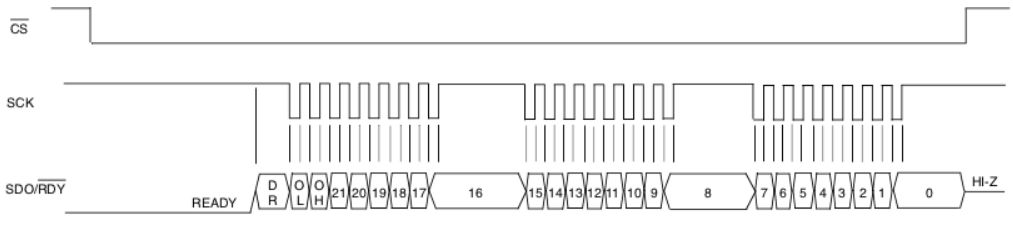


Figure 4-2: Summary of output coding data format. This shows example digital output codes for specific analog input voltages when using a typical SPI communication. [REF]

Once the RPi has read the ADC value, this value is then converted to a decimal value using a binary-to-decimal conversion. The output voltage of the sensor is a value between 0 and 5V. Since the range of decimal values is $-2{,}097{,}152$ to

2,097,151 , the decimal value d needs to be converted to a fraction and then multiplied by 5V to get the voltage output value V_{IN}.

$$V_{IN} = \frac{d + 2097152}{4194304} \cdot V_{REF}, \quad V_{REF} = 5V \tag{4.8}$$

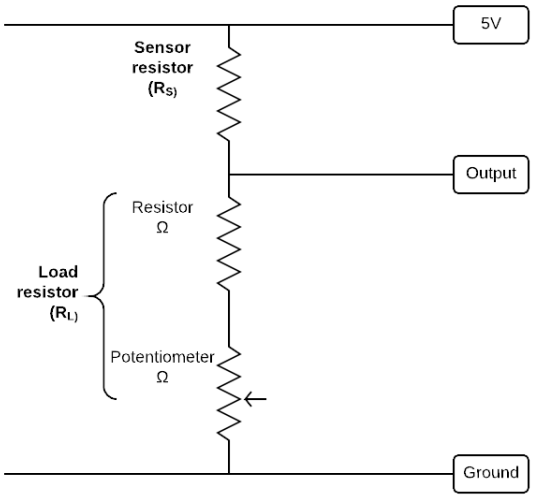


Figure 4-3: Circuit diagram for MQ-series sensors. This shows how the sensor uses R_S and R_L to make a voltage divider. [EDIT]

Using the calibration curves on the MQ131 sensor data sheet (Figure 4-5a), it is possible to calculate an approximate ozone reading in $\mu g/m^3$. The first part of the calculation is based on the circuit structure (Figure 4-3) and requires the resistance of the sensor (R_S) in clean air (R_0). Unfortunately, there are no controlled ozone environments available at the university for calibrating the sensor. Therefore, we assume that the indoor air was clean and contained minimal ozone. To calculate R_0, set the sensor's potentiometer to its minimum resistance (0Ω) and record the output voltage V_L. Then, also set the potentiometer to its maximum resistance ($10k\Omega$) and record the output voltage V_H. Figure 4-4 shows the potentiometer of both the MQ131 and MQ135 sensors. Once the output voltage at these two points

Figure 4-4: MikroElektronika click board potentiometers. (L) blue potentiometer of ozone click (MQ131); (R) blue potentiometer of air quality click (MQ135).

has been recorded, it is possible to calculate R_0 by using the following equation:

$$V_{OUT} = V_{IN} \cdot \frac{R_L}{R_0 + R_L}, \qquad V_{IN} = 5V \tag{4.9}$$

which can be rearranged to

$$R_0 = (\frac{V_{IN}}{V_{OUT}} - 1) \cdot R_L \tag{4.10}$$

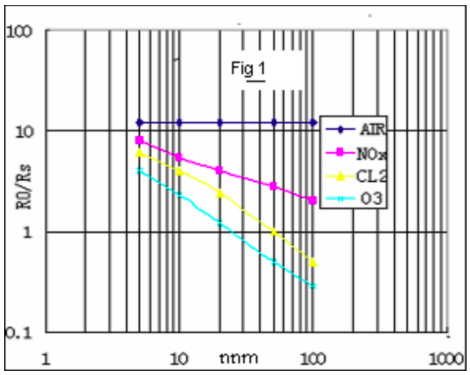
In fact, there are two equations since we have two different pairs of R_L and V_{OUT}.

$$R_0 = (\frac{5}{4.934} - 1) \cdot 10100 = 135.103, \quad R_0 = (\frac{5}{2.015} - 1) \cdot 100 = 148.139 \tag{4.11}$$

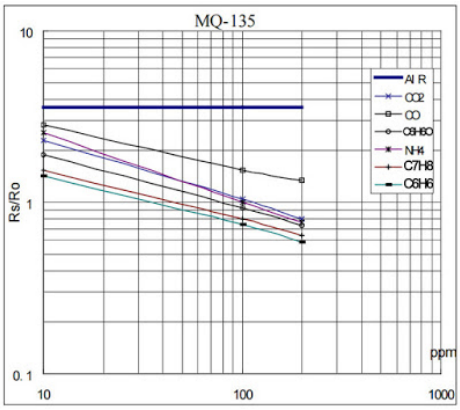
Theoretically, these calculated R_0 values should be the same, but here we take the average as they differ slightly (likely due to their low cost).

$$\therefore R_0 = \frac{135.103 + 148.139}{2} = 141.6 \tag{4.12}$$

Since we now know R_0, we can set the potentiometer so the output voltage is reasonable (i.e. around a mid-point) and then work out the corresponding load



(a) MQ131.



(b) MQ135.

Figure 4-5: Calibration curves from sensor data sheets. (a) shows the calibration curve for the MQ131 sensor, with ozone being the light blue line, and (b) shows the calibration curve for the MQ135 sensor, with a number of different sensitivities.

resistance R_L:

$$R_L = \frac{V_{OUT} \cdot R_0}{V_{IN} - V_{OUT}} \tag{4.13}$$

Then in any environment, it is possible to calculate R_S:

$$R_S = (\frac{V_{IN}}{V_{OUT}} - 1) \cdot R_L \tag{4.14}$$

This enables the calculation of the ratio R_S/R_0, which can be inputted into the calibration curve (Figure 4-5a) to output an approximate ozone reading in $\mu g/m^3$. We digitise the calibration curve to obtain (x, y) pairs and then use power regression to obtain the final equation:

$$O_3(ppb) = (0.0541 \frac{R_S}{R_0})^{(-\frac{1}{0.917})} \tag{4.15}$$

4.3 General air quality sensor

The MQ135 air quality sensor also outputs a raw analog signal that represents voltage. To avoid repetition, only the differences in the ADC chip used are explained here, with the rest of the process following that described above for the

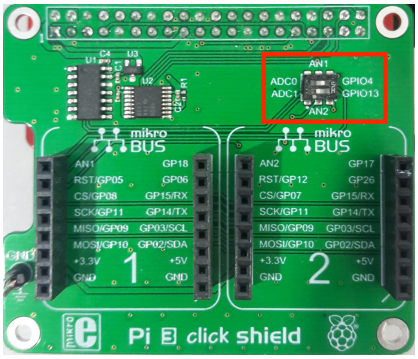


Figure 4-6: GPIO/ADC switch. The switch is highlighted in red on the Mikro Pi 3 Click Shield HAT. This switch enables the signal to either be sent directly to the GPIO pins or directed to the ADC first.

MCP3551 chip.

The Air Quality Click from MikroElektronika did not contain a dedicated ADC chip, unlike the Ozone Click. Therefore, a Mikro Pi 3 Click Shield HAT is used, which provides a GPIO/ADC switch so the relevant option can be selected. The switch for the Ozone Click is set to GPIO since it has an in-built ADC, whilst the switch for the Air Quality Click is set to ADC, thus routing the sensor's output through the ADC (see Figure 4-6). The Pi 3 Click Shield uses an MCP3204, which is a 4-channel 12-bit ADC chip. As with the MCP3551, the communication with the RPi uses SPI and each data word is made up of 3 bytes. Initiating communication is also achieved by bringing the $\overline{CS}$ low. However, they differ in their data transfer protocol. The RPi (SPI master) must first send configuration bits to the MCP3204: 5 leading zeroes, a start bit (1) and then 4 configuration bits. The ADC then samples the analog signal for 1 clock cycle, before responding with 1 null bit and then the data over the next 12 bits, MSB first.

These 12 bits allow for a range of digital values between 0 and 4095. Again, the decimal output d from the ADC needs to be converted to a fraction and then multiplied by 5V to get the voltage output value V_{IN}.

$$V_{IN} = \frac{d}{4096} \cdot V_{REF}, \quad V_{REF} = 5V \tag{4.16}$$

A similar calibration process can be attempted, but using the calibration curve in Figure 4-5b instead. However, this graph does not detail NOx on any of the

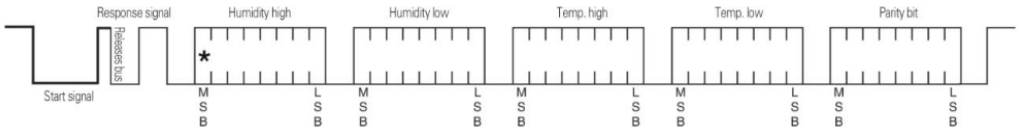


Figure 4-7: DHT22 communication protocol. The data word is made up of 40 bits: 16 bits for the humidity reading, 16 bits for the temperature reading and the final 8 bits as a check-sum.

calibration curves, despite the sensor description detailing it is sensitive to NOx. Upon further research, it was found that the MQ135 sensor is actually sensitive to a number of different gases that indicate poor air quality, including ammonia, nitrogen oxides, benzene, smoke, carbon monoxide and carbon dioxide. However, it is impossible to isolate any of these gases from the signal without further equipment. Therefore, much of the analysis herein for this sensor focuses on the raw voltage output and only the voltage delta can be analysed, with no precise NOx relation. Essentially, the voltage acts as a 'basket' measure of air quality.

4.4 Temperature and humidity sensor

The output of the DHT22/RHT03 temperature and humidity sensor is a digital signal. The sensor uses a single-bus for communication, where the sensor is the slave and the computer is the master. Each data word is made up of 40 bits: 16 bits for the humidity reading, 16 bits for the temperature reading and the final 8 bits as a check-sum. The humidity and temperature readings are split into two parts: the first byte is the integer part and the second byte provides the fractional part. To read from the DHT22, the RPi must send a read command (pull low for $18\mu s$, pull high for $20 - 40\mu s$) and then the sensor replies with an acknowledgement (pull low for $80\mu s$, pull high for $80\mu s$). The data transfer then follows, with $50\mu s$ low for a 0 and $70\mu s$ high for a 1. The data is sent MSB first. Binary-to-decimal conversion is used to convert the binary values to temperature readings in Celsius and humidity values in percentage form.

4.5 Limitations

One of the shortcomings of our calibration process for the Shinyei PPD42 particulate sensor is that it assumes particle shape and size. It could be possible to use pulse-width modulation (PWM) to record the size of particles that pass in front of the sensor's detection beam, although this may be inaccurate due to the sensor's inability to isolate individual particles. Despite the increase in cost, other particle detection methods, such as gravimetric measurement, may provide more precise measurement of particulate concentrations.

Furthermore, there are no known concentrations of ozone gas available at the university, meaning we cannot measure the output voltage of the MQ131 sensor at specific concentrations of ozone. As a result, we cannot plot points at different concentrations to create our own calibration curve. Instead, we are forced to use the data sheet's calibration curve, which may be inaccurate.

Similarly for the MQ135, we do not have access to known concentrations of NOx in the sensor's detection range. We did manage to test the sensor output using 500 ppm NOx gas in the university's Automotive Research Group laboratory. However, this concentration is far greater the normal range of NOx found in the atmosphere and so overwhelmed the sensor. Moreover, the aforementioned issue relating to the sensor's cross-sensitivities means we cannot calibrate the voltage to a specific concentration of a single gas. This means the device cannot identify locations of high NOx, but nonetheless is still useful in identifying poor general air quality.

Moreover, because these sensors are low-cost, they have cross-sensitivities with other atmospheric gases that can potentially cause unwanted changes in sensor output. However, without the correct testing equipment or more expensive sensors that only have very minimal cross-sensitivities, it is not possible to correct for these effects in the time frame of this project. This is unfortunately a common issue when using low-cost sensors, hence their price.

It is also worth noting that the data sheets for these sensors indicate that their outputs can drift over time, which has also been found by previous studies [REF]. Our data collection period is 2 weeks, which is too short to notice a change as they typically occur over bi-annual or annual periods of time. However, this effect should be tested and accounted for if the device is to be used for longer periods

of time.

Chapter 5

Methodology

Chapters 3 and 4 detailed how the air quality monitoring device, and system as a whole, was constructed and how the sensors were calibrated to output meaningful values. These chapters collectively described the process undertaken in creating the final device, providing a product that can now be used to answer our research questions.

This chapter details the methodology behind how we applied this device to answer our main research questions. Each method is described in detail, with our reasoning provided in each case to support our choices.

We begin by restating our three main research questions. Then, we provide the rationale behind our project location choice and describe the data cleaning and validation process that will be applied to each dataset collected. The following three sections then focus on each of our research questions in turn, detailing the method we used and the data collection process in each case.

5.1 Research questions

As Chapter 2 concluded, there are a number of gaps in the current literature surrounding air quality monitoring devices. Using these gaps, we narrow the focus of this study into answering three main research questions. These are as follows:

1. Can air pollution be monitored to a satisfactory level using a bicycle-based monitoring device? (Section 5.4)

2. Can such a device improve spatial and temporal resolution of air quality measurements compared to fixed monitoring stations? (Section 5.5)

3. How can these air quality monitoring devices be applied for public benefit? (Section 5.6)

By attempting to answer these questions, we will advance the literature in bicycle-based air quality monitoring and provide a solid platform for future development of a compact and portable monitoring device.

5.2 Location

Given this project is being conducted at the University of Bath, we carry out the data collection in the city of Bath, UK. In addition to the proximity to the university, the city is also chosen for its traffic and pollution problem, which has led to the creation of the Bath Air Quality Action Plan (BANES, 2017). This plan, created by the Bath and North East Somerset (BANES) council, sets out how an improvement in the air quality in Bath will be achieved between 2017 and 2022. This document is a requirement from the government, following BANES being identified by the government as 1 of 23 local authorities 'with persistent exceedances required to undertake local action to consider the best option to achieve statutory NO_2 limit values within the shortest possible time' (DEFRA, 2017b). By collecting air pollution data in Bath, the outputs of this project could help the council better understand pollution in Bath by comparing the council's own fixed monitoring stations with the mobile measurements from this project. This may also identify previously unknown problem pollution areas.

Furthermore, Bath is geographically interesting because of its position surrounded by steep hills. These hills act as a channel for wind coming from the west and could potentially direct air pollution into Bath, as well as trapping it in the city. Figure 5-1 shows an elevation raster map of Bath, with the colouring and shading indicating the terrain and altitude.

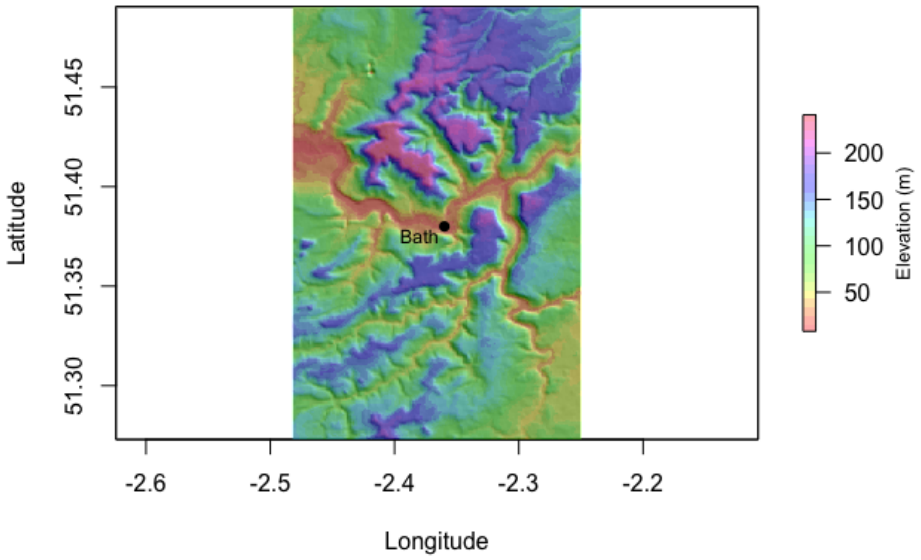


Figure 5-1: Elevation raster of Bath. This image shows how Bath is surrounded by steep hills to the north, east and south, which act as a funnel for wind and, therefore, air pollution into Bath.

5.3 Data cleaning and data validation

Whilst ideally our sensors will not output anomalous readings, it is almost inevitable given their low cost. Therefore, it is prudent to detail our approach to cleaning the data outputted by these sensors, especially for future reproducibility.

Data cleaning is the process of transforming raw data into consistent data that can be analysed. The operations carried out influence the statistical properties of the data, meaning they should be conducted with caution and in a reproducible manner.

Outlier detection is an important part of the data cleaning process. Outliers are often detected by considering the distribution of values around a mean μ with standard deviation σ and an indicator of the confidence interval z:

$$\mu \pm z \times \sigma \tag{5.1}$$

Values that fall outside this range are unlikely to be from the underlying

distribution and so can be considered outliers.

However, the mean and standard deviation are not robust to outliers (i.e. the addition of one large erroneous value can disproportionately influence these values). Given we are using low-cost sensors that are more prone to errors than industry-grade sensors, a robust detection method is needed. As a simple check for the presence of outliers, one can examine the distribution of values using a histogram. If some bars are spaced considerably far away from the majority of values, this can indicate outliers are present.

In order to more precisely identify which measurements are outliers in our data, we use a three-part approach. In each case, we apply the outlier detection method and if all of these methods report a reading to be an outlier, it is filtered out from the sample through the use of a type of band pass filter.

1. Inter-quartile range (IQR) (Section 5.3.1)
2. Median and Median Absolute Deviation (MAD) (Section 5.3.2)
3. S_n (Section 5.3.3)

5.3.1 IQR

The IQR was created by Tukey (1977) and provides a measure of statistical dispersion. It is defined as the difference between the 25th and 75th percentiles. Coupled with the 25th and 75th percentiles (Q_1 and Q_3, respectively), it can be used to define an outlier detection rule. Values outside the range shown in Equation 5.2 are considered outliers.

$$[Q_1 - 1.5 \times IQR, \quad Q_3 + 1.5 \times IQR] \tag{5.2}$$

5.3.2 Median and MAD

The median can be used as a measure of central tendency and is considered to be more robust than the mean, with a breakdown point of 50% (the proportion of erroneous observations an estimator can handle before giving an incorrect result). This is, in fact, the highest possible breakdown point. The median is defined as the value that separates the distribution into two equal halves.

The median absolute deviation (MAD) can be used as a measure of the variability in the data and is more robust than the standard deviation, with a breakdown point of 50%. The MAD is calculated by finding the median of absolute deviations about the median. More formally, it is defined as follows:

$$MAD = b \cdot M_i(|x_i - M_j(x_j)|) \tag{5.3}$$

where $M(\cdot)$ is the median function, x is an observation and b is a constant scale factor depending on the distribution.

These two values can then be used to identify outliers using the following statistic:

$$\frac{|x_i - M_j(x_j)|}{MAD} \tag{5.4}$$

for each x_i and flagging those x_i which exceed a certain cutoff (typically 2.5 or 3).

5.3.3 S_n

The MAD takes a symmetric view on dispersion, meaning it can perform poorly on asymmetric distributions (i.e. skewed). Rousseeuw and Croux (1993) suggested an alternative estimator, S_n.

$$S_n = c \cdot M_i(M_j|x_i - x_j|) \tag{5.5}$$

Equation 5.5 is very similar to Equation 5.4, except that the median function $M_j(\cdot)$ has been moved outside the absolute value. The breakdown point of estimator S_n is still 50%, but it has a better efficiency than MAD.

5.3.4 Time series

Since our dataset is a collection of time series, the outlier detection should also take this into account. There are two ways we incorporate this into our methodology.

Firstly, the outlier detection techniques detailed above are also run using moving averages. The dataset is separated into distinct collection runs (e.g. YYYY-MM-DD AM) and the moving average outlier detection is run on each of these. The reasoning for using a moving average is to account for any trends in the pollution data. For example, if, on a main road, the pollution level is much higher

than the average for that collection run, it would not make sense to use the outlier detection methods on the dataset as a whole as otherwise these higher values could be removed erroneously (i.e. false positive).

The second way we account for the time series nature is by using a similar method to van Zoest, Stein and Hoek (2018). The authors appear to be one of the first to create a method for identifying outliers in spatio-temporal air quality data. Their method consists of separating the data into spatial and temporal divisions. For example, urban traffic and urban background (spatial), and weekdays/weekends and traffic times (temporal). The reason for this was that air quality can depend highly on the local area and the time of day, so outlier detection must take these divisions into account. As a result, the spatio-temporal variability in pollution concentrations is maintained. Our data cannot be split easily into spatial categories as we focus on a small city, but we can separate the data into 4 temporal categories. The first split is for a weekday or weekend, the rationale being that traffic is highly likely to be higher during the week when people are commuting to and from work. The second split is for the first half of the day (AM) and the second half of the day (PM). When combined, these separations produce four categories:

1. Weekday AM
2. Weekday PM
3. Weekend AM
4. Weekend PM

5.3.5 Geo-spatial outliers

It is also sagacious to have a process in place to identify and correct geo-spatial outliers i.e. anomalous locations of measurements. The GPS module is relatively low-cost and if it loses a strong signal from the satellites then the locations may be anomalous. Plotting these values on a map could lead to inaccurate conclusions being drawn.

The geo-spatial outliers are corrected by extrapolating from the closest known prior location. We calculate the bearing and estimate the corrected location by

moving along that bearing by the average distance travelled every 10 seconds on the bicycle. Whilst this method makes some strong assumptions, the average distance travelled is only 49.7 metres, which means any deviation from the actual route will be minimal.

5.4 Question 1: Can air pollution be monitored to a satisfactory level using a bicycle-based monitoring device?

In order to answer this research question, we need to analyse three distinct parts. Firstly, we need to collect data along a consistent route for a suitable period of time that includes varying conditions. By analysing the sensor outputs under these different conditions, we can test whether the device can identifying trends/differences in air pollution. Secondly, different positions of the device on a bicycle need to be considered. Finally, the optimal method of inducing air flow over/into the air quality sensors needs to be analysed.

5.4.1 Route selection

The selected route is based upon the current fixed air quality monitoring stations in Bath, so that measurements obtained by this project can be compared to the official council measurements. The locations of these stations are displayed and labelled in Figure 5-2.

The cycling route passes by all of these stations, and attempts to follow the main roads where possible as these likely have the highest levels of pollution. This route has been overlaid on the above figure, with the resulting map shown in Figure 5-2.

Furthermore, BANES outlined an 'Air Quality Management Area' in 2013. This is shown in Figure 5-3. Again, the chosen cycling route almost completely falls within this area, thus strengthening our rationale behind the route's choice.

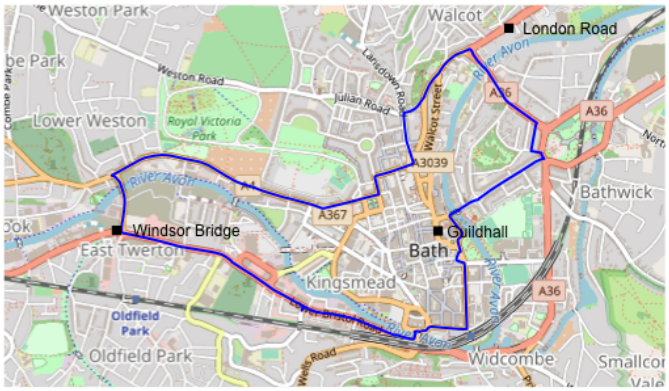


Figure 5-2: Map of fixed air quality monitoring sites in Bath with the chosen cycling route overlaid. The route passes by all three fixed monitoring stations.

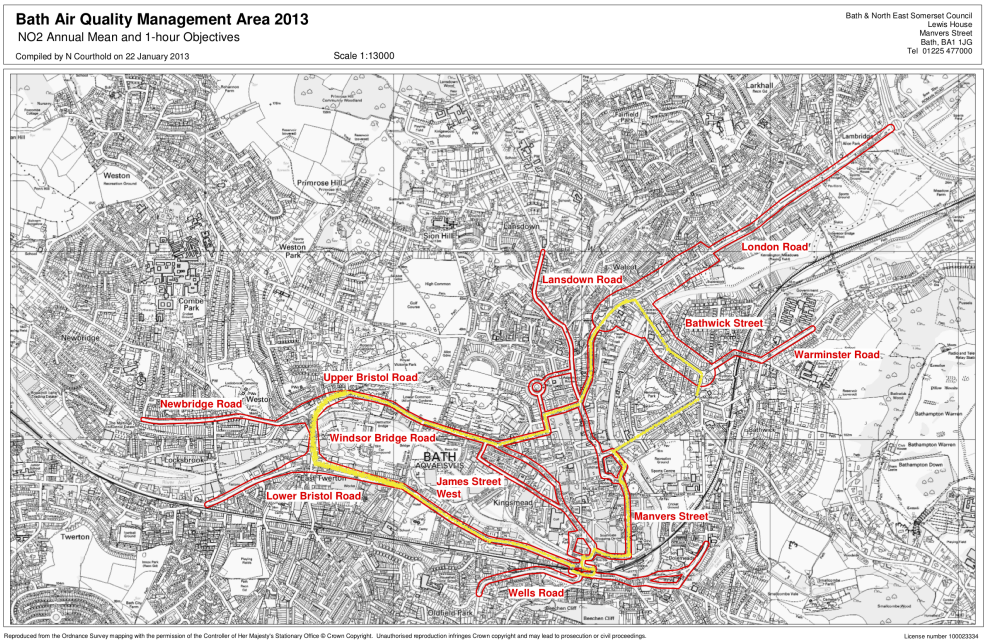


Figure 5-3: Bath Air Quality Management Area (BANES, 2017) with shaded cycling route.

5.4.2 Data collection

To collect data, we ride a bicycle with the air quality monitoring device attached around a pre-determined route, detailed in Section 5.4.1. The route is traversed in both directions in order to test whether there are any differences in pollution depending on direction of travel and to help identify anomalous measurements. This leads to minor differences in the route due to certain one-way streets in Bath. We ride the bicycle at a relatively constant speed in order to distribute measurements regularly in spatial terms.

Data is collected across 3 weeks in July and August, from Monday 23rd July to Tuesday 14th August. The route is cycled both in the morning and afternoon. The morning run is to coincide with the morning rush hour and the evening run is to coincide with the evening rush hour. These times are deemed to have the highest pollution levels across the day. 3 weeks is also deemed long enough to allow for weather variations, which allows us to analyse the effect of weather on the measurements.

5.4.3 Pollution mapping

With the pollution data collected along the route detailed in Section 5.4.1, we can create pollution maps that show how pollution varies along the route on average, at different times of day and at different times of the week. To achieve this, we overlay hexagonal shapes on a map based upon the method used by Hoang et al. (2013). Each hexagon's colour is an average of all values that fall inside it's boundaries. Additionally, Smallbone (2012) found that the public prefer a blue or green colour for representing good air quality and red for representing poor air quality. Consequently, we also follow this colour scheme.

5.4.4 Positioning on bicycle

In order to identify which position on a bicycle is optimal for an air quality monitoring device, we need to test the device attached to different positions. We identify three potential positions on a bicycle, shown in Figure 5-4. Position 1 is situated directly below the rider and could potentially be impacted by particles thrown up by the front wheel. It is also not as visible to the user. On the other

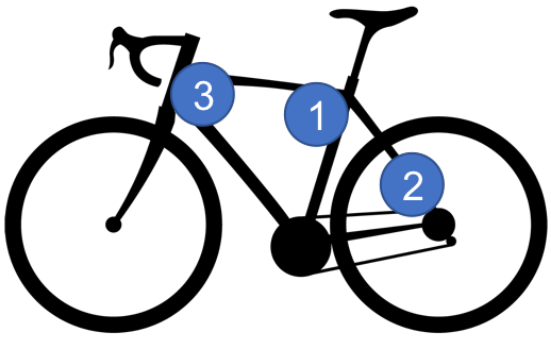


Figure 5-4: Potential positions for device on bicycle. (1) is above the front wheel, (2) is directly below the rider, and (3) is on the seat stays.

hand, this position may provide good protection from the elements and help to make the device robust. Position 2 is the least visible, but is situated lowest on the bicycle frame and, thus, provides a different testing environment for the sensor. This may also be impacted by particles thrown up by the wheels. Finally, Position 3 is situated above the front wheel, thus avoiding any particles thrown up by the wheels. In addition, it is the most suitable position for the user to view the device and allows the user to notice when the system-status LED is flashing.

We cycle with the device attached in each position in turn along the same stretch of road. This testing is also carried out across a short time period in order to minimise the effect of changes in the environmental conditions. The results are then compared to ascertain whether the position of the device makes a difference, analysing whether any major spikes occur in the data, which could indicate that the device is being impacted by particulate matter thrown up by the wheels.

5.4.5 Air flow testing

As Section 2.6.2 detailed, air flow can affect the sensor measurements. In particular, a light-scattering particulate detecting technique, which the Shinyei PPD42 sensor uses, relies on sufficient air flow to pass particles through a beam of light. As a result, our air flow testing focuses on the particulate sensor. We propose three different air flow systems for this device:

1. Sensor exposed directly to air – air flow will come naturally from motion of

bicycle

2. Fan directs air at the sensor
3. 3D printed funnel directs air flow into the sensor

Similarly to the position testing in Section 5.4.4, we will cycle with the device attached along the same stretch of road, changing the air flow system each time. The results will be compared to see which air flow system provides the lowest variability in measurements.

5.4.6 Robustness

There is no empirical test conducted in this project to measure the robustness of the air quality monitoring device, since such a test would be a small project in itself. Instead, two factors are analysed: (i) whether the device is impacted by the weather, and (ii) whether the device is impacted by the vibrations of the bicycle.

5.5 Question 2: Can this device improve spatial and temporal resolution of air quality measurements compared to fixed monitoring stations?

In order to answer this research question, we firstly need to demonstrate how the device can collect measurements across a greater area compared to a fixed monitoring station, whilst not sacrificing temporal resolution (Section 5.5.1). By collecting measurements across a larger area, it is possible to generate a more detailed view of pollution in an area, thus improving spatial resolution. Secondly, we need to demonstrate how improved temporal resolution is possible without sacrificing spatial resolution (Section 5.5.2).

5.5.1 Locus around fixed monitoring station

A fixed monitoring station can only take measurements in exactly one location. Whilst they do typically use high-quality, expensive monitoring equipment that offers higher accuracy and reliability, they have poor spatial resolution. Our

bicycle-based air quality monitoring device is portable and so can take measurements at varying locations with ease. To demonstrate this improvement in spatial resolution, we need a suitable location and data collection method.

Location

As Section 5.4.1 details, there are three fixed air quality monitoring stations in Bath. Of these three, only the Bath Guildhall site is suitable for a locus-based collection. The Windsor Bridge location (see Figure 5-5a) is surrounded by industrial estates and adjacent to a bridge over the river in Bath, hence the station's name, and is, therefore, unsuitable as it there are an insufficient number of roads to cycle around. The London Road location (see Figure 5-5b) is also situated next to the river and is located at the foot of an extremely steep hill with 16.6% gradient. Thus, it does not have a suitable number of roads in a locus around the station and so is also unsuitable. The Bath Guildhall is the only monitoring station remaining and, although it is situated near to the river, it is closest to to the city centre and so there are numerous roads in its vicinity and on the opposite side of the river. This site is shown in Figure 5-5c.

Data collection

For the data collection, a circular locus is drawn around the monitoring site with a radius of 350 m and a route created that covers as much of the locus as possible in the time frame allowed. One difficulty here is that Bath has a relatively high number of one-way streets, which means the route is constrained by these. The Bath Guildhall station reports NOx readings every 15 minutes, meaning our route is chosen so that it takes approximately 15 minutes to cycle around, finishing back at the origin each time. The selected route is shown in Figure 5-6. By undertaking this part of the experiment, we are be able to compare our readings against the readings reported by the high-quality fixed monitoring station, as well as demonstrate an improved spatial resolution without sacrificing temporal resolution.

(a) Windsor Bridge

(b) London Road

(c) Guildhall

Figure 5-5: Road network around fixed monitoring stations. (a) is surrounded by industrial estates and a river, with few suitable roads. (b) is sided by a river and a steep hill, so the number of suitable roads is also limited. (c) is next to the river, but there are numerous roads around the site and on the opposite side of the river.

Demonstrating higher spatial resolution

The collected data is used to create a higher resolution pollution map than can be generated from a single fixed monitoring station. The measurements are first plotted as points on a map to demonstrate how a larger area can be monitored. We then use Kriging/IDW interpolation to generate an even higher resolution pollution map. The results from these are detailed in Section 6.2 in the next chapter.

5.5.2 Event route

One novel application of a mobile air quality monitoring device is that it can quickly be deployed to monitor the air quality around an event's location. For example, it can be deployed to a large sporting event that significantly impacts

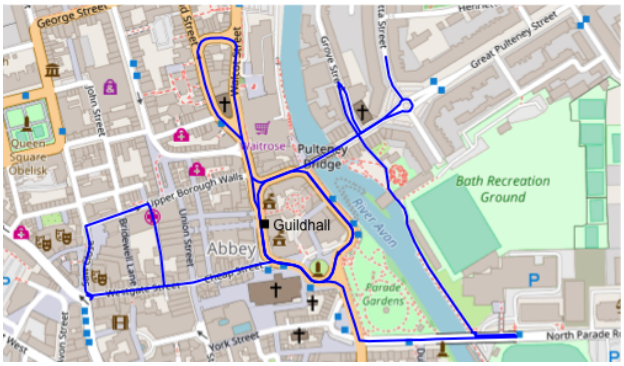


Figure 5-6: Locus-based route around Guildhall fixed monitoring station. This route covers a sufficiently large area around the site in order to demonstrate improved spatial resolution, whilst maintaining the temporal resolution of measurements (each lap takes 15 minutes).

the traffic in an area in order to analyse the effect of that event on the air quality. An extensive literature search did not discover any previous studies that have applied portable air quality monitoring to specific events. Therefore, we believe we are the first to study this application.

Location

A football match in Bath, with a ground capacity of approximately 8000 people, is a sizeable event likely to draw a large crowd and significantly increase traffic in the city. We did consider a Bath Rugby match, which would have likely drawn a larger crowd. However, the rugby stadium is situated nearby the railway station and does not have its own carpark, meaning a large proportion of the spectators would likely travel via train, which is unlikely to impact the pollution in Bath significantly. The football stadium is situated in the west of Bath, away from the railway station, and has its own car park. We believe this event is more likely to cause a significant change in pollution.

Data collection

For the data collection, we chose a straight route along the high street, which is situated just outside of the football stadium and car park. People travelling by car to the game will either have to park in the car park, or on the road the

Figure 5-7: Event-based route outside of Bath City FC football stadium. The route traverses the high street back and forth, which is likely to experience the greatest increase in traffic compared to other roads in the vicinity.

stadium. Therefore, this chosen route should experience the greatest increase in traffic from the event compared to other roads in the area.

To test the event-based capabilities of the device, we collect air quality measurements from along the route before, during and after the sporting event, thus allowing comparison of air quality across these three distinct time periods. Due to short length of the route, we walk with the device instead of cycling. In each time period, we continuously walk back and forth along the route, which takes approximately 5 minutes in each direction.

Demonstrating higher temporal resolution

The data provides a pollution level along the route for every 5 minute interval, which we then use to show how the pollution changes over the whole time period. This more granular picture of air quality over time demonstrates the improved temporal resolution that this device can offer, as well as proving its use in monitoring events whilst maintaining portability. The results of this are shown in Section 6.2 in the next chapter.

5.6 Question 3: How can these air quality monitoring devices be applied for public benefit?

There are a variety of route planning services offered by companies. For example, Google Maps, Waze, Citymapper and TomTom. These services take a number of different factors into account in order to calculate the optimal route from origin to destination. Such factors include distance, journey time, traffic and fuel consumption. However, as a result of the negative health impacts of pollution detailed in Section 2.1.1, the public have become increasingly concerned about air pollution and wish to minimise their exposure to pollution (De Hartog et al., 2010).

However, due to the lack of high spatial resolution pollution data, particularly on a town or city level, it has been difficult to create a routing service that takes air pollution into account. This project is one of the first to attempt such routing (Sharker and Karimi, 2014; Hasenfratz et al., 2015) and, thus, provides a step towards improving public health through an alternative routing application.

To create such an application, complete pollution data is needed for an area. We recognise this is infeasible for a singular cyclist to collect and so we analyse the feasibility of commuters providing this data for us. We then convert Bath's road network into a suitable graph for route finding and invent a novel cost function that generates weights for each road of the network. Finally, we apply a search algorithm that finds the lowest cost path from an origin to a destination.

5.6.1 Using commuting cyclists to collect pollution data

Bath's roads have a total length of 52.8 km, as calculated by the `osmnx` python module using OpenStreetMap data (see Section 5.6.2 for further discussion of this module). Given the resource constraints of this project, it is infeasible to collect pollution data across the whole city whilst only using one cyclist. Moreover, collecting any more than one time period of data would be even more difficult. Consequently, we simulate the pollution data across Bath. We section the city into a raster of 100 by 100 cells, assigning a pollution value to each cell.

However, whilst this simulation of data is sufficient for this project, we want to understand how commuting cyclists could be used in a city to collect the

pollution data, which could then be fed into the routing algorithm. The foremost question here is how many cyclists are needed to generate sufficient coverage of a city. To test this, we generate random origin and destination pairs, plotting the shortest route between the two points. From these random routes, we calculate the percentage of a road network covered. This is calculated for number of cyclists $n = 1$ up to $n = 500$.

Using the most recent census data (Office for National Statistics, 2011), the urban area of Bath has a population of 94782 and 1.9% of the population cycle to work. This means there are an estimated $94782 \times 1.9\% = 1801$ people that commute via bicycle in Bath every day. Clearly, not all of these commutes will be within the boundaries of our test area, so testing up to $n = 500$ is not an unreasonable figure.

In order to minimise the number of trivial routes, we introduce a minimum commute distance of 500 m. This avoids the origin and destination being extremely close to each other.

The calculated coverage percentages are then plotted on a graph to visualise how the coverage of the city changes with the number of commuting cyclists.

5.6.2 Road network graph

To create a graph that represents the road network in Bath, we use the python module `osmnx` (Boeing, 2017), which uses data from OpenStreetMap. It is possible to specify the mode of transport when creating the graph, with the choices being walking, cycling and driving. We chose driving, despite the data being collected on a bicycle. The reason for this was that cycling commuters prefer to use bike paths (Broach, Dill and Gliebe, 2012; Skov-Petersen et al., 2018), of which there are few in Bath, meaning cyclists will typically cycle on the roads.

We represent Bath as a *directed* graph $G = (V, E, W_d, W_p)$, consisting of a set of vertices V, a set of edges E and two sets of edge weights W_d and W_p. We herein refer to vertices as 'nodes'. Each node $v_i \in V$ represents an intersection between two or more roads, or a dead-end. Each edge $e_j \in E$ represents a road between two nodes and is associated with weights $w_{j,d} \in W_d$ and $w_{j,p} \in W_p$, where $w_{j,d} = d_j$ is the length of the edge and $w_{j,p}$ is a weight generated by our cost function. The graph generated by `osmnx` for Bath is shown in Figure 5-8.

Figure 5-8: Graph produced using `osmnx` for Bath, UK. Each edge of the graph (grey line) represents a road segment and each node (blue dot) represents a function or road-end.

We make two assumptions here. Firstly, we assume that the length of a road $w_{j,d}$ is proportional to the time it takes to traverse the edge e_j from n_i to n_{i+1}. Secondly, we assume that a longer travel time and/or an increase in altitude along an edge e_j leads to an increased inhalation of pollution. Despite there being exercise-related health benefits of cycling, the physical nature means cyclists breathe deeper and more frequently, thus increasing the amount of pollutants inhaled (Bigazzi and Figliozzi, 2014; Ramos, Wolterbeek and Almeida, 2016). This can offset the health benefits of the exercise, meaning cyclists have an incentive to travel along less-polluted streets.

We then combine these assumptions with three variables we believe will affect the health impact of travelling along a road. The distance of the road, from our previous assumption, clearly impacts how long an individual is exposed to the pollution, so we include the product of the length and the pollution level. This product was used as a cost function by Hasenfratz et al. (2015), which is shown in Equation 5.6. The routes generated by this cost function decreased pollution

exposure by 7.1% on average.

$$w_{p,j} = \sum_{k=1}^{n} d_{j,k} \cdot c_k \tag{5.6}$$

where n is the number of $100m \times 100m$ grid cells the edge penetrates, $d_{j,k}$ is the length of edge e_j penetrating grid cell k, and c_k is the modelled pollution concentration in grid cell k.

We expand upon this relatively simple cost function by accounting for two additional variables. Firstly, we include whether the road has a positive or negative gradient, which is a variable that is specific to human-powered methods of travel (i.e. driving a car has no physical exertion). This has an impact because cycling uphill is physically more demanding, which means the cyclist will be breathing more deeply and, thus, inhaling more pollution in any given time frame. This effect is reversed when cycling downhill as the cyclist's physical exertion is likely to be lower. We introduced this to our cost function as a factor: a positive gradient g_j implies $1 + g_j > 1$ and a negative gradient implies $1 + g_j < 1$. Secondly, each edge e_j of our graph has a road classification $type_j$ associated with it, which provides a proxy for the traffic level. For example, a 'primary' road will likely have higher levels of traffic, which could slow down the cyclist and lead to increased pollution exposure times. We assigned a value to each classification and added this to the distance product. Our final cost function is, therefore, as follows:

$$w_{p,j} = (1 + g_j) \cdot (d_j \cdot p_j + type_j) \tag{5.7}$$

Since data collection for this application is deemed infeasible within the project's constraints, the data is generated by sampling points from a distribution. The distribution of pollution measurements in the main route dataset was used to make these values as realistic as possible.

In order to assign the relevant pollution value to each edge for input into the cost function, we calculated the centre point along each road and used the k-nearest neighbours (KNN) algorithm to find the closest k values (we used $k = 5$). The KNN algorithm implemented Haversine distance to account for the curvature of the earth's surface. We then averaged these values for each road. This method is visualised in Figure 5-9. Whilst using only the centre of each road may be a

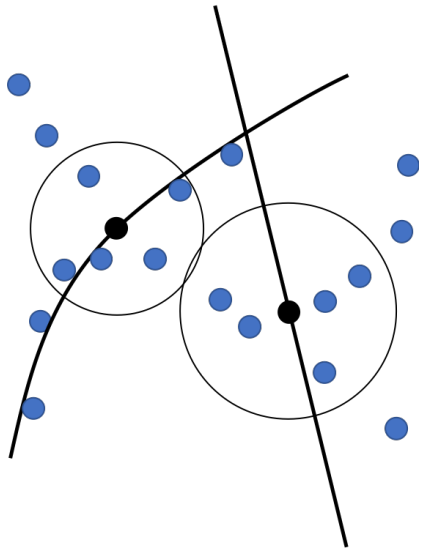


Figure 5-9: Example of selecting k-nearest air quality measurements for two roads (k = 5).

strong simplification, many of the roads are not straight and so are separated into multiple segments by `osmnx`, meaning the centre points are still representative of their road segments.

5.6.3 Search algorithm

The graph comprises of $|V| = X$ nodes and $|E| = X$ edges, where an edge represents a road and a node represents an intersection of two or more roads. We use Dijkstra's algorithm (Dijkstra, 1959) as our search algorithm for this application. It finds the lowest cost path between any two nodes in a graph. In this case, the two nodes are the origin O and destination D. This algorithm starts by expanding the origin node O and analyses the available edges. It takes the edge with the lowest cost each time and then analyses each edges from the resulting node, building up a set of known distances from O to each visited node. The algorithm stops once it has visited the destination node.

The aforementioned minimum commute distance was also implemented here to avoid short trips that are unlikely to have any deviation between the shortest and health-optimal routes.

5.6.4 Comparing routes

Once the required number of 'health-optimal' and shortest route pairs has been generated, we compare these routes to analyse the percentage decrease in pollution exposure and the percentage increase in distance. The results of this are detailed in Section 6.3 in the following chapter.

Chapter 6

Results

Using the methodology detailed in Chapter 5, this chapter presents the results obtained from the air quality monitoring device. The chapter is structured in the order that the research questions were detailed in the methodology. In each case, we detail the the results obtained and the statistical significance, if relevant.

6.1 Question 1: Can air pollution be monitored to a satisfactory level using a bicycle-based monitoring device?

Across the data collection period, 9877 measurements were collected along the chosen cycling route. In this section, results are presented for assessing whether this device can monitor pollution to a satisfactory level. First, the results for the positioning of the device on the bicycle are shown. Second, the data quality is assessed and any outliers removed. Third, this data is used to create pollution maps of Bath, which are analysed to understand how the pollution level changes along the route. Next, any observable patterns in the measurements are presented and, finally, results demonstrating the robustness of the device are shown.

6.1.1 Positioning on bicycle

Figure 6-1 displays the results from testing the air quality monitoring device in different positions on the bicycle, with a picture of each position underneath the

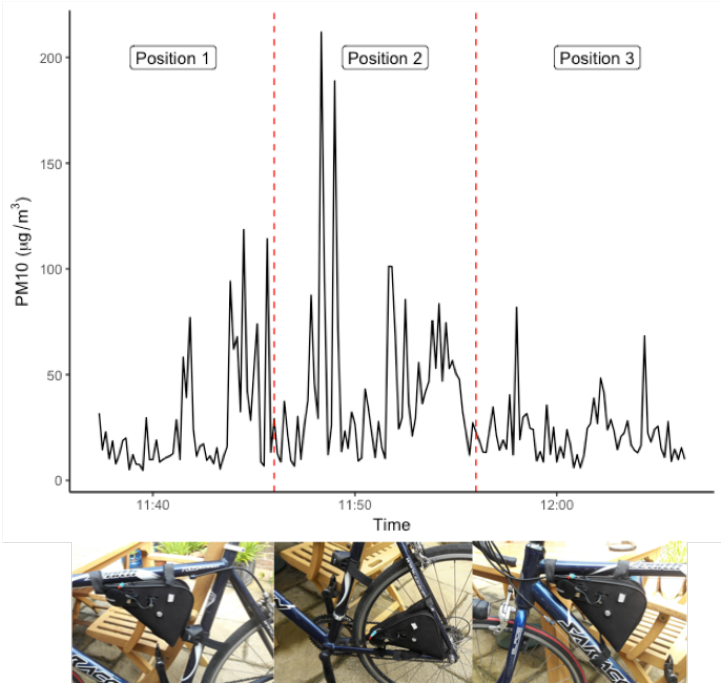


Figure 6-1: Bike position testing. The device was attached in 3 different positions on the bike. Positions 1 and 2 have large spikes in their measurements, whereas position 3 is relatively stable.

corresponding section of the graph. The results are displayed for PM10 only as the O_3 and general air quality measurements show no significant difference in pattern.

Overall, the figure shows varying levels of PM10 across the three positions. Position 1 and Position 2 have relatively large spikes in their measurements compared to the baseline of approximately 20-25$\mu g/m^3$, with Position 2 experiencing the highest levels of PM10. Position 1 has two readings over 100$\mu g/m^3$, whilst Position 2 has two readings close to the 200$\mu g/m^3$ level. Conversely, Position 3 appears to be the most stable position, with lower variation in the measurements compared to the other two positions.

Table 6.1 displays the mean and standard deviation for each of the three positions. Position 2 has the highest mean and the highest standard deviation, whereas Position 1 has the lowest mean and standard deviation.

Table 6.1: Summary statistics by device position on bicycle.

	PM10 ($\mu g/m^3$)			
Position	Mean	Std. dev.	Max	Min
1	26.70	27.62	118.58	4.79
2	43.44	38.67	211.99	6.63
3	21.81	13.62	81.86	5.93

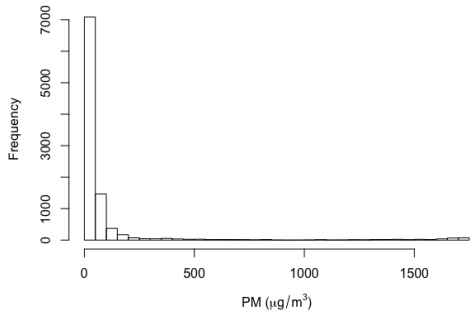
6.1.2 Data quality

Firstly, it must be mentioned that half-way through the data collection the MQ131 and DHT22 sensor broke due to moisture exposure. The MQ131 sensor was replaced and calibrated using the same method as detailed in Section 4.2. The DHT22 sensor was also replaced, although it required no calibration. The readings from the period during which these sensors were broken were discarded. Furthermore, the particulate sensor occasionally outputs a zero reading, but a zero concentration of particulates is extremely unlikely. Therefore, we also treat these measurements as errors and exclude them prior to conducting the outlier detection so they do not skew the detection ranges.

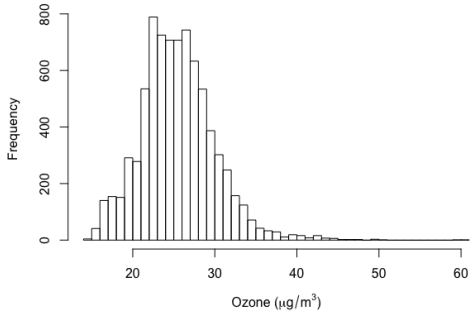
We first conduct exploratory data analysis to understand the quality of data collected and identify where potential outliers may lie. We then implement robust outlier detection methods to identify outliers, which are then excluded from our investigation. A different technique is then employed to identify and remove geo-spatial outliers. By understanding the quality of the data from this air quality monitoring device, it will help to show whether this device can satisfactorily monitor air pollution.

Exploratory analysis

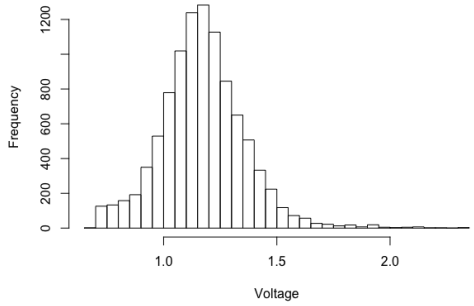
As Figure 6-2a shows, the distribution of PM10 measurements is positively skewed and it appears that the measurements contain a large number of outliers. The majority of the values are distributed between 0 and 200, with a number of other values spread out to over 1500. Within the frequent observations, the positive skew indicates that lower measurements tend to be more common than higher measurements. Compared to the other histograms, the PM10 measurements from



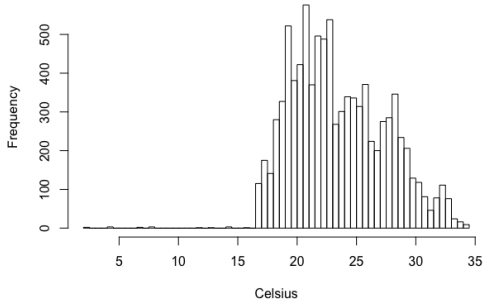
(a) Shinyei PPD42



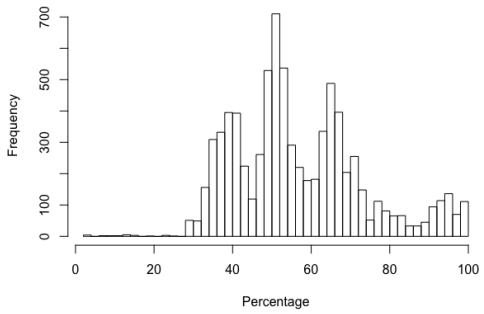
(b) MQ131



(c) MQ135



(d) DHT22 temperature



(e) DHT22 humidity

Figure 6-2: Histograms depicting the distribution of pollution sensor measurements. (a) shows a positive skew, with a number of extreme values; (b) and (c) have approximately normal distributions; (d) has a slight positive skew, with low frequencies of values below 17 °C; (e) also has low frequency values below 25% humidity, with a more varied distribution of values compared to the other histograms.

the Shinyei PPD42 particulate sensor have the largest range of values and, consequently, potentially the largest proportion of outliers.

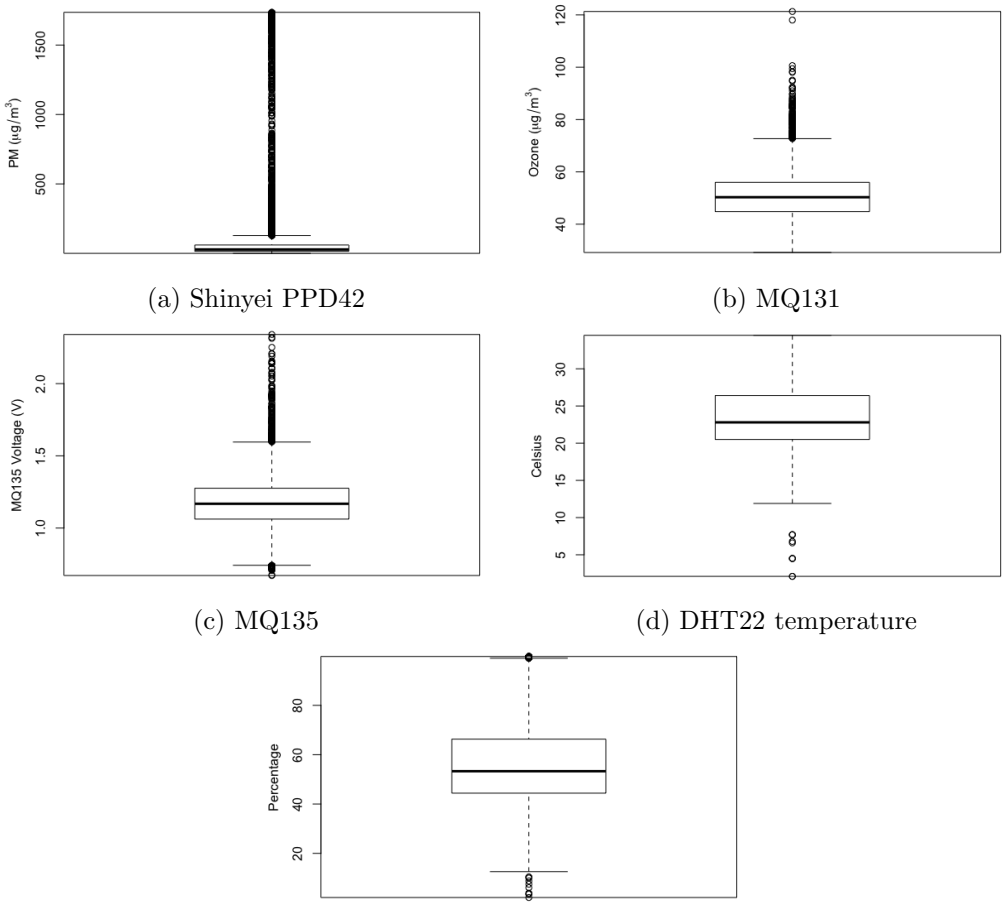
Figures 6-2b and 6-2c show approximately normally distributed values from the MQ131 and MQ135 sensors, with some of the higher values potentially being outliers due to the long, thin tail on the right-hand side of the distribution.

The measurements from the DHT22 temperature and humidity sensor do not follow a normal distribution as closely. The temperature measurements (Figure 6-2d) have a slight positive skew, with a range of low frequency values below 17 °C. Likewise, the humidity measurements (Figure 6-2e) have a number of low frequency values below 25% humidity, suggesting a number of outliers are present. These are noticeable by the sudden drop in frequency to the left-hand side of the distribution. Overall, the distribution of humidity measurements is more erratic, with a number of peaks in the distribution across a larger range.

IQR

The first method for detecting outliers, as detailed in Section 5.3.1, uses the IQR. The IQR is visualised in Tukey's boxplot by the height of the box – the top indicates the 75th quartile and the bottom indicates the 25th quartile. Figure 6-3 shows the boxplots for each of the sensors. These show that all of the sensor measurements have a non-zero number of outliers, although the Shinyei PPD42, MQ131 and MQ135 sensors appear to have the most, indicated by the number of points outside of the 'whiskers' of the boxplots. These whiskers represent the outlier detection rule of 1.5 times the IQR away from the 25th and 75th quartiles.

The results of performing this outlier detection method are shown in Table 6.2. It is clear from the table that the PM10 measurements have the highest percentage of outliers (11.75%) compared to the ozone (1.79%) and general air quality measurements (1.97%). The upper bound of 128.64 $\mu g/m^3$ is reasonable, given that the highest PM10 measurement recorded in 2017 by the Windsor Bridge monitoring station in Bath was 204 $\mu g/m^3$ (Courthold, 07/08/2018). The MQ131 sensor has the lowest percentage of outliers when using the IQR detection method. It must be noted that the sample size N for the ozone measurements is considerably lower than for the other two pollutants due to the aforementioned breakage and replacement of the MQ131 sensor midway through data collection.



(a) Shinyei PPD42

(b) MQ131

(c) MQ135

(d) DHT22 temperature

(e) DHT22 humidity

Figure 6-3: Tukey's box-plot of sensor measurements. The 'whiskers' represent the outlier boundaries. (a) displays a highly skewed distribution with a large number of outliers at the high-end of the distribution; (b) and (c) have similar distributions with a number of outliers at the high-end; (d) only has a small number of outliers at low values and is otherwise evenly distributed; (e) also has an even distribution with a small number of outliers at both high and low values.

Table 6.2: Outlier detection summary using IQR method on the whole dataset. The particulate measurements have the highest percentage of outliers.

Gas/particulate	N	Median	IQR	U.B.	L.B.*	# outliers	%
PM10 ($\mu g/m^3$)	9089	28.13	45.50	128.64	0.00	1068	11.75
O_3 ($\mu g/m^3$)	7920	50.31	11.17	72.74	28.06	142	1.79
General AQ (V)	9877	1.17	0.21	1.60	0.74	195	1.97

*Lower bound (L.B.) capped at 0 as negative concentrations are nonsensical.

Table 6.3: Outlier detection summary using median and MAD method on the whole dataset. The outlier percentage is higher for particulates compared to using the IQR method, but lower for the ozone and general air quality measurements.

Gas/particulate	N	Median	MAD	U.B.	L.B.*	# outliers	%
PM10 ($\mu g/m^3$)	9089	28.13	25.38	104.26	0.00	1285	14.14
O_3 ($\mu g/m^3$)	7920	50.31	8.30	75.21	25.42	105	1.33
General AQ (V)	9877	1.17	0.16	1.65	0.69	139	1.41

*Lower bound (L.B.) capped at 0 as negative concentrations are nonsensical.

However, the percentage of outliers is similar to that for the general air quality measurements, so this exclusion does not appear to have detrimentally impacted the analysis from a statistical perspective.

Median and MAD

The median and MAD outlier detection method was also implemented. Outliers were identified as any measurements which were more than 3 times the MAD away from the median.

The results of this method are detailed in Table 6.4. Once again, PM10 shows the highest percentage of outliers, with 14.14% of the measurements being considered outliers. This is a 2.39% increase over the number of outliers found by the IQR detection method. On the other hand, there has been a small decrease in the outlier percentages for the ozone and general air quality measurements. However, overall, the values are not too dissimilar to those found by the IQR method, which indicates the robustness of these methods.

Table 6.4: Outlier detection summary using median and S_n method on the whole dataset. The outlier percentage is higher for particulates compared to using the IQR method, but lower than the MAD method.

Gas/particulate	N	Median	S_n	U.B.	L.B.*	# outliers	%
PM10 ($\mu g/m^3$)	9089	28.13	25.22	103.80	0.00	1288	14.17
O_3 ($\mu g/m^3$)	7920	50.31	8.46	75.69	24.94	97	1.22
General AQ (V)	9877	1.17	0.17	1.67	0.67	121	1.23

*Lower bound (L.B.) capped at 0 as negative concentrations are nonsensical.

S_n

Table 6.4 shows the result of the S_n method of outlier detection. The PM10 outlier percentage is only 0.07% less than using the MAD method. The percentage of outliers for ozone and general air quality are the lowest out of the three methods.

Per time period group

The above methods are also implemented on a per-group basis, with the group being the week period (Week AM, Week PM, Weekend AM, Weekend PM). These results are shown in column 2 of Table 6.5. In the interest of clarity, the table is simplified to only show the main figures. The results show that, on average, running the outlier detection on a per-group basis leads to a slightly decreased number of measurements identified as outliers. Although the number of outliers rises when using the MAD and S_n methods, the number of outliers detected by using the IQR method actually falls. It is also clear that PM10 has the highest number of outliers. When running these methods on the sample as a whole, the MAD method leads to the highest total number of outliers, but this changes to the S_n when running the methods per time period grouping.

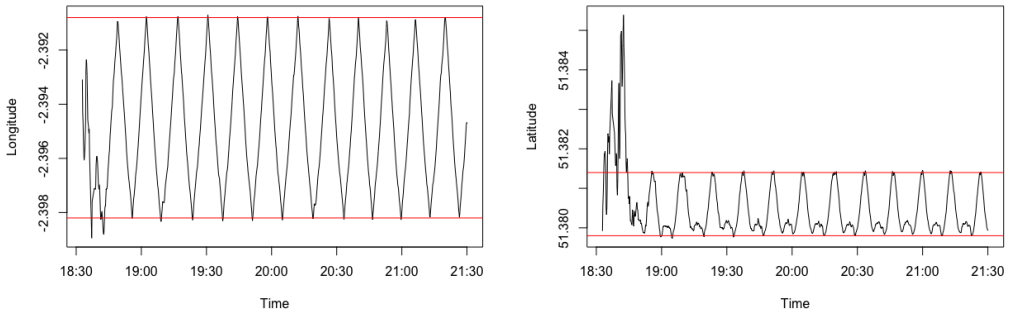
Moving average

The outlier detection methods are also applied using moving averages, to account for the time series nature of the data. The results are shown in column 3 of Table 6.5. The results show that the number of outliers for PM10 reduces considerably, but the figure increases for O_3 and General AQ measurements, with the number

of outliers increasing by approximately a factor of 3.

Table 6.5: Total number of outliers when running each method on the whole sample or per time period grouping. On average, per time period grouping leads to a higher number of measurements being identified as outliers.

	Whole dataset				Time period grouping				Moving average			
Gas/particulate	IQR	MAD	S_n	Avg.	IQR	MAD	S_n	Avg.	IQR	MAD	S_n	Avg.
PM10	1068	1285	1288	1214	1026	1298	1304	1209	980	800	769	850
O_3	142	105	97	115	136	89	93	106	612	502	463	526
General AQ	195	139	121	152	201	152	141	165	674	406	342	474
Total	1405	1529	1506	1480	1363	1539	1538	1480	2266	1708	1574	1850



(a) Longitudes (b) Latitudes

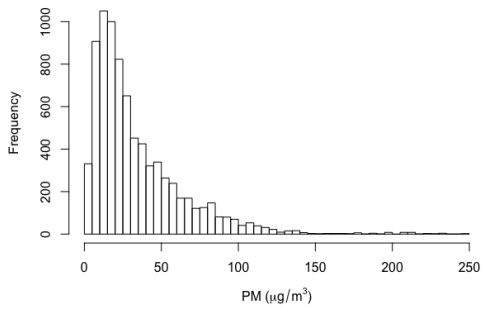
Figure 6-4: Longitude and latitude data from a consistent route creates a waveform, which can be used to identify outliers. Values outside of the pattern's minimum and maximum (shown by the horizontal red lines) or that appear irregular can be considered outliers.

Geo-spatial outliers

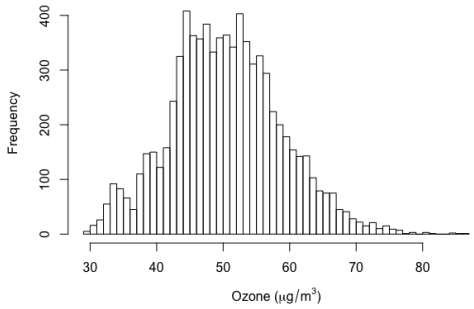
There were a small number of geo-spatial outliers, which were corrected using the method detailed in Section 5.3.5. Since the routes in each of the three data collection periods are consistent, the longitude and latitude measurements create waveform-like patterns. These are useful in identifying outliers as they clearly show when the value exceeds the usual minimums and maximums (represented by red horizontal lines). For example, Figure 6-4 shows the longitude and latitude time series from the event-based collection period. Figure 6-4a has erratic longitude measurements until 18:45, before settling into a consistent pattern. Figure 6-4b has a slightly longer period of inconsistent latitude measurements and the values are clearly erroneous as they stretch far away from the peaks of the pattern.

Summary

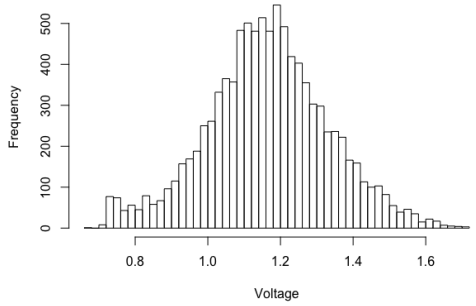
The use of moving averages when applying these methods results in too many measurements being identified as outliers – many of these would otherwise not be considered outliers if the methods were run on the dataset as a whole. We think it is beneficial to separate the dataset into time period groupings, however. Therefore, the methods are run per time period grouping. For a value to be



(a) Shinyei PPD42



(b) MQ131



(c) MQ135

Figure 6-5: Histograms depicting the distribution of pollution sensor measurements with cleaned data. (a) shows a slight positive skew; (b) and (c) have approximately normal distributions.

considered an outlier, all of the three methods – IQR, MAD and S_n – have to identify it as an outlier. These values are then filtered out. Histograms of the pollution measurements from the cleaned dataset are shown in Figure 6-5. The resulting changes in dataset sizes, from the initial dataset to the final dataset with outliers removed, are shown in Table 6.6.

Regarding the geo-spatial outliers, these have been corrected as best possible. Where they still remain erroneous, these measurements have been discarded from any geo-spatial analysis. However, the geo-spatial measurement being an outlier does not affect the pollution sensor measurements, so these remain for the other

Table 6.6: Dataset size changes.

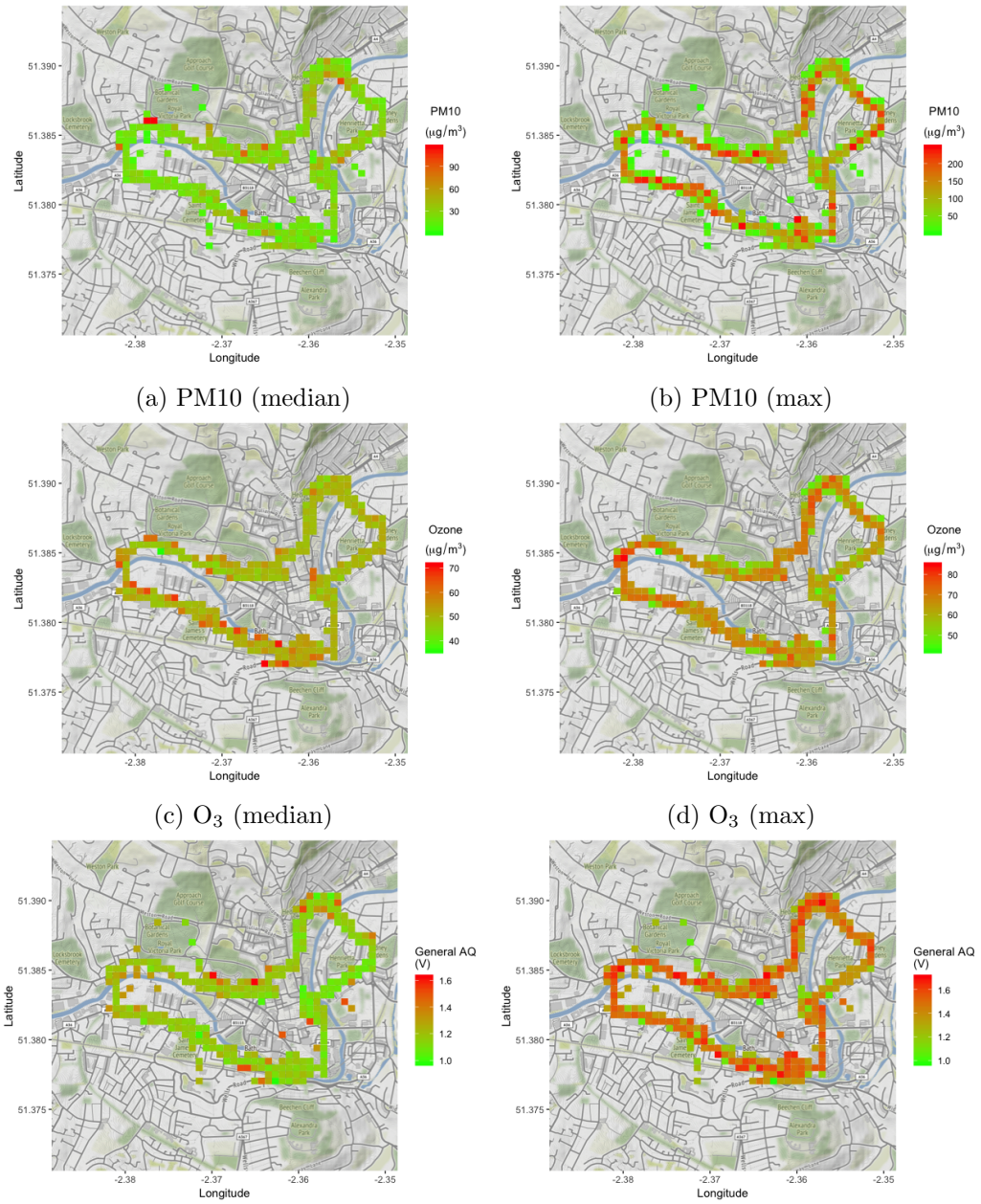
	Initial	Errors removed		Outliers removed	
Gas/particulate	N	N	Change	N	Change
PM10	9877	9089	788	8063	1026
O_3	9877	7920	1957	7784	136
General AQ	9877	9877	0	9676	201
Total	29,631	26,886	2745	26,030	856

analyses.

6.1.3 Maps

Using the cleaned dataset, pollution levels along the route can be mapped. Figure 6-6 shows pollution maps for each of the pollutants. Each pollutant has two maps: the first uses the mean of all measurements inside each tile's boundaries to colour the tile and the second uses the maximum. By analysing the colours of the tiles in these maps, it is possible to identify high pollution areas in Bath.

Figure 6-6a has a high level of pollution in the north west corner of the route. This location is directly outside the waste recycling facility in Bath, which may be the source of the pollution. By using the maximum, Figure 6-6b shows that large sections of the route have measured a high level of pollution at some point during the data collection period. For the average ozone levels in Figure 6-6c, the worst pollution is along the road in the south west part of the route. This road is Lower Bristol Road and is one of the busiest roads in Bath because it leads to the A36 (a major road). Figure 6-6d does not show much variation in the maximum pollution levels around the route. The only noticeable maximum in solid red is again in the north west of the route, on the north side of Windsor Bridge. This is located at one of the larger traffic junctions in Bath and is situated nearby the aforementioned waste recycling centre, both of which could be contributing to this higher reading. Figure 6-6e does not have any identifiable problem areas for general air quality, with any high points being sparsely dotted around the route. Taking the maximum measurements in each tile, the majority of the route has experienced a high measurement from the general air quality sensor.



(a) PM10 (median)

(b) PM10 (max)

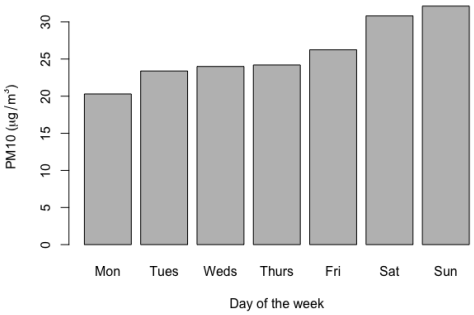
(c) O_3 (median)

(d) O_3 (max)

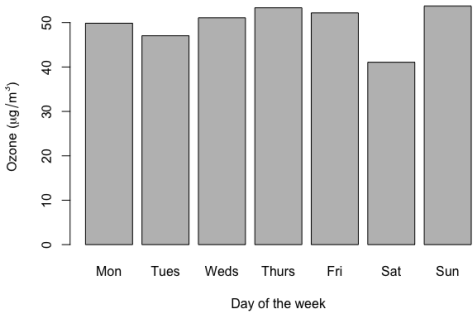
(e) General AQ (median)

(f) General AQ (max)

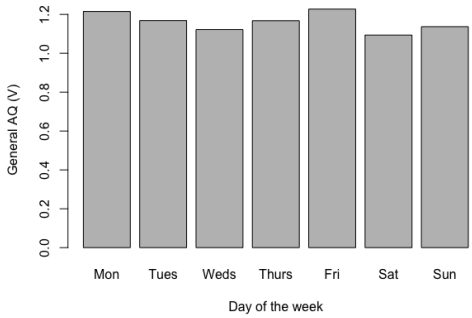
Figure 6-6: Air quality pollution maps of the route. The maps are separated vertically by the type of pollutant, with the LHS showing the median values and the RHS showing the maximum values.



(a) PM10



(b) O_3



(c) General AQ

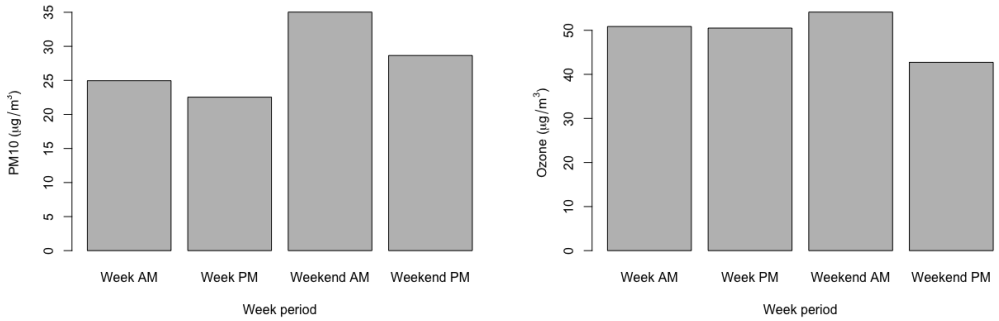
Figure 6-7: Pollution by day of the week. Each bar represents the median pollution level for that day. (a) shows PM10, (b) shows O_3, and (c) shows general air quality.

6.1.4 Pattern analysis

In this section, any patterns that can be identified in the data collected by the air quality monitoring device are displayed.

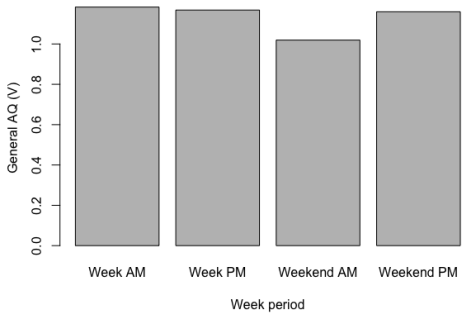
Figure 6-7 shows the median pollution levels for each pollutant by the day of the week. As Figure 6-7a shows, the average pollution level increases throughout the week, from a low on Monday of $20\mu g/m^3$ to a high on Sunday of over 30 $\mu g/m^3$. There is no identifiable pattern in Figure 6-7b for O_3 or Figure 6-7c for general air quality.

Figure 6-8 shows the median pollution levels for each pollutant by the week period, as initially detailed in Section 5.3.4. For PM10 and O_3, the AM measure-



(a) PM10

(b) O_3



(c) General AQ

Figure 6-8: Pollution by period of the week. Each bar represents the median pollution level for that day. (a) shows PM10, (b) shows O_3, and (c) shows general air quality.

ments are on average higher or equal to the PM measurements. This relationship does not hold for the general air quality measurements, which have a higher average for Weekend PM compared to Weekend AM.

6.1.5 Robustness

One issue encountered during the data collection was sensors breaking due to exposure to moisture. It rained during two of the data collection periods, which led to the ozone sensor (MQ131) and the humidity sensor (DHT22) ceasing to work. The MQ131 sensor's output consistently decreased until it reached zero, around

which it remained. The DHT22 sensor's reading increased to 99.9% humidity and did not reduce again. The sensors were left on in a dry environment to try and extract the moisture, but this did not solve the problem.

A second issue was the device losing power. It is thought that severe vibrations, caused by something such as riding over a pothole in the road, led to the micro-USB cable from the external battery to the RPi losing its connection. The parts of the device were not secured in the triangular bicycle bag, which may have led to this issue occurring.

6.2 Question 2: Can this device improve spatial and temporal resolution of air quality measurements compared to fixed monitoring stations?

6.2.1 Bath Guildhall locus

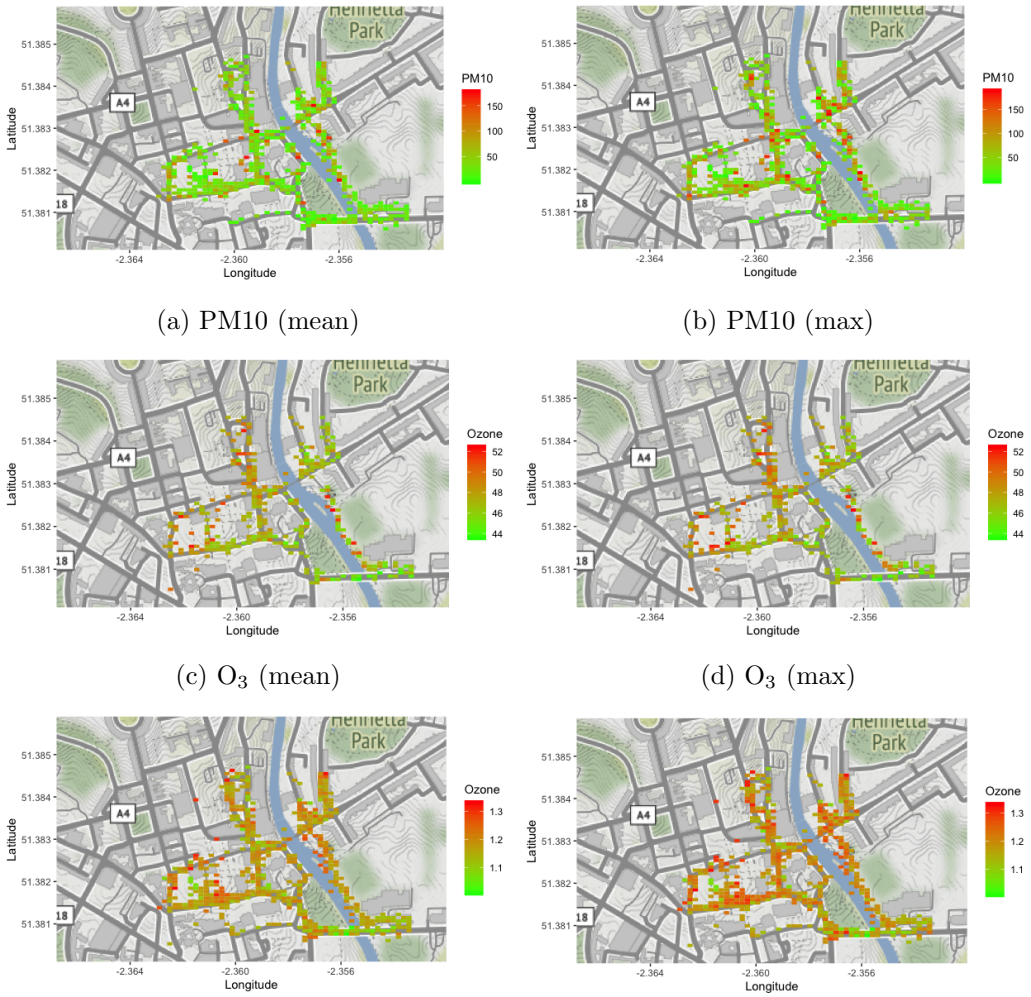
Using the chosen route within the locus around the Guildhall in Bath, 1452 measurements were collected on a Friday afternoon. Using the same outlier detection and removal technique detailed in Section X, the dataset was cleaned. Since the data was collected in one afternoon period, the dataset could not be separated into temporal categories for the outlier detection, as was previously done with the route dataset. Instead, the outlier detection methods were run on the dataset as a whole. This led to XXX outliers identified and removed.

Figure 6-9 displays the pollution maps of the locus route for the three pollutants.

6.2.2 Bath football event

1044 measurements were collected for the Bath City FC football match. As with the locus dataset, the data was cleaned by running the outlier detection methods on the dataset as a whole. X outliers were detected and removed.

Figure 6-10 visualises the pollution along the route outside the football stadium over time. The vertical lines represent key time intervals: the first line marks the start of the football match and the second marks the end. There is a heatmap for each pollutant. The x-axis represents time bins for every complete



(a) PM10 (mean)

(b) PM10 (max)

(c) O_3 (mean)

(d) O_3 (max)

(e) General AQ (mean)

(f) General AQ (max)

Figure 6-9: Pollution maps of locus route. (a) and (b) show PM10; (c) and (d) show O_3; (e) and (f) show general air quality.

one-way traversal of the route and the y-axis represents longitude bins for each section of the route (the route has been separated into 20 longitude bins).

For PM10, Figure 6-10a shows a relatively high level of pollution prior to the match (indicated by the high proportion of red tiles), which then decreases as the time approaches kick-off (19:45). The pollution level remains low for the duration of the match and does not change significantly after the match has finished. The

graphs for O_3 and the general air quality are not so clear. Figure 6-10b does not show as much variation. There is a 'green' period just before the match starts, but then the pollution level rises again after kick-off. There is an increase as the match ends, although it is not entirely consistent along the route. For the general air quality, Figure 6-10c there is some variation in the measurements, with a noticeable increase at the end of the game.

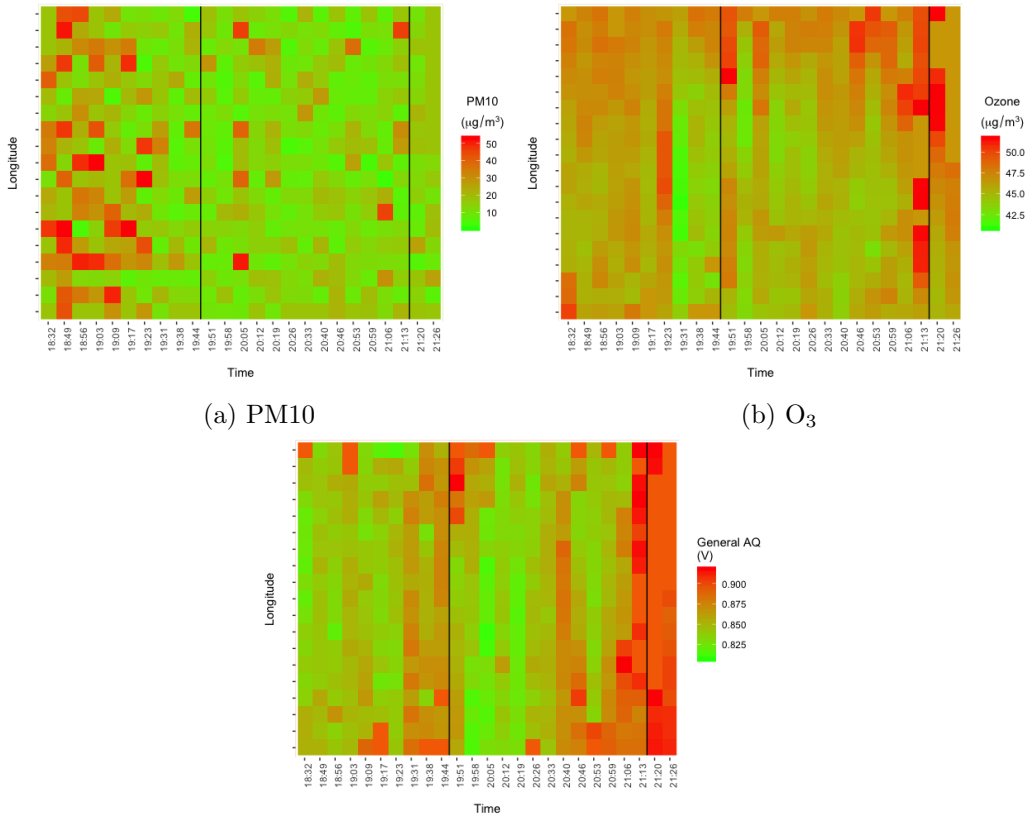
Furthermore, each vertical column of this graph represents a complete one-way traversal of the route outside the football match, taking approximately 7 minutes each time. This demonstrates that the device can satisfactorily measure pollution across a small spatial area with an improved time resolution compared to one of the fixed air quality monitoring stations in Bath (7 minutes compared to 15 minutes). If the device were to be moved less, then the temporal resolution could be increased further, up to a maximum temporal resolution of 10 seconds (the time interval between taking sensor measurements) if the device were stationary.

6.3 Question 3: How can these air quality monitoring devices be applied for public benefit?

Having shown that this air quality monitoring device can measure air pollution to a satisfactory level and can improve spatio-temporal resolution, we now demonstrate how it can be used to reduce the public's exposure to air pollution.

6.3.1 Coverage

As expected, the commuter simulations show an increase in coverage of the city as the number of cyclists increases. There are diminishing returns to an increase in cyclists, however. Interestingly, over 50% of the roads in Bath can be covered with 50 cyclists. With 500 cyclists, it is possible to obtain 90.3% coverage. Whilst it may appear odd that the other 10% is missing [re-word], Figure X shows that these unvisited roads are likely to be dead-ends or residential cul-de-sacs, which are not of much importance in the analysis.



(a) PM10

(b) O_3

(c) General AQ

Figure 6-10: Heatmaps of event pollution. (a) shows PM10, (b) shows O_3, and (c) shows general air quality.

6.3.2 Comparison of shortest and health-optimal routes

We found that by taking the health-optimal route instead of the shortest route, one can decrease their pollution exposure by 10.3% on average. However, this comes at the cost of a 13.5% increase in distance travelled, on average.

It is worth noting that the reduction in pollution exposure can greatly depend on the given origin and destination, as illustrated in Figure 6-12. For example, in Figure 6-12a there is no difference between the shortest route and the health-optimal route. On the other hand, Figure 6-12c shows large deviations from the shortest route, with the health-optimal route being 57.1% longer.

Furthermore, these findings must be taken with caution. If more people began

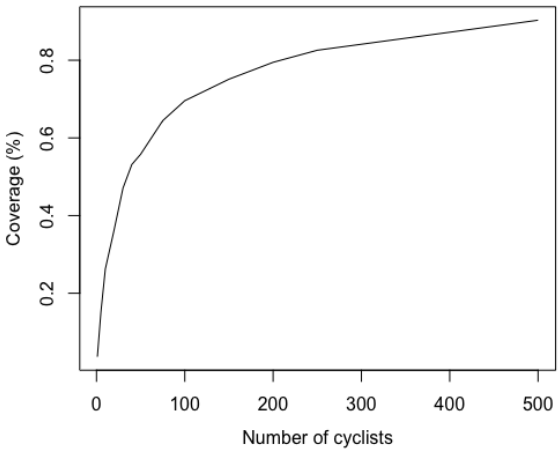


Figure 6-11: Percentage of Bath theoretically covered by n cyclists. Over 50% of the city can be covered with only 50 cyclists, with diminishing returns to scale as the number of cyclists increases.

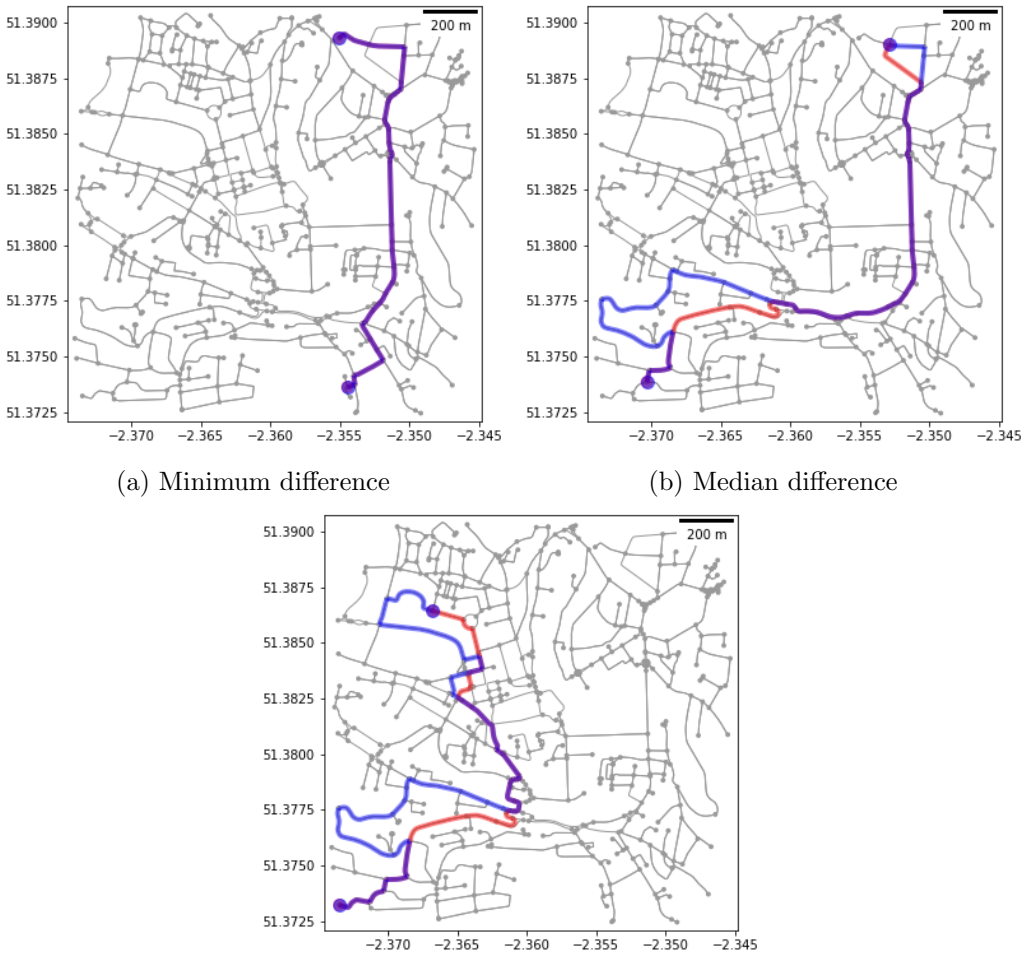
Table 6.7: Health-optimal route pollution reductions and length increases

Gas/particulate	Pollution change	Length change
PM10	-10.3%	+13.5%
O_3	-X%	+X%
General air quality	-X%	+X%

to use these health-optimal routes, they may cease to be health-optimal routes by the very nature of more people travelling along and polluting them. Therefore, there is a clear need for this application to be dynamic. What we mean by this is the routing should use real-time pollution data (or as close as one can get to real-time) and base the current health-optimal route on this data, thereby avoiding a surge in traffic along one particular route.

6.3.3 Health benefit

Choosing the health-optimal route can lead to a substantial reduction in pollution exposure. We now quantify the health benefit associated with this pollution



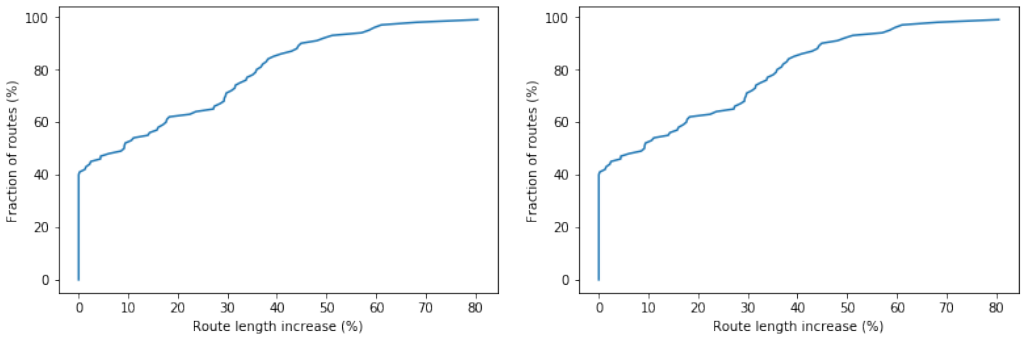
(a) Minimum difference

(b) Median difference

(c) Maximum difference

Figure 6-12: Three example origin-destination pairs in Bath with the corresponding shortest route (red) and health-optimal route (blue). (a) shows no difference between the routes, whereas (b) shows some deviation and (c) shows a large amount of deviation.

reduction, based on previous findings.



(a) Reduction in pollution exposure.

(b) Increase in route length.

Figure 6-13: Cumulative depictions of the route length increases and pollution exposure decreases from the 500 generated routes. A sizeable percentage of routes show no increase/decrease, even with a minimum route length of 500 m. There are then diminishing number of routes as the delta increases.

Chapter 7

Discussion

7.1 Question 1: Can air pollution be monitored to a satisfactory level using a bicycle-based monitoring device?

No baseline to compare against. But maps do show variation, so we are measuring something.

Given this was carried out over a short time period and along the same quiet stretch of road, it was not expected that the environmental conditions would change significantly over the collection period. Therefore, the results should be relatively constant. However, the figure shows that Position 1 and Position 2 have relatively large spikes in their measurements. These positions are underneath the rider and on the seat stay (refer to Figure **??** for images of these positions) and were identified as positions that could potentially be impacted by particulate matter thrown up by the wheels.

7.2 Question 2: Can this device improve spatial and temporal resolution of air quality measurements compared to fixed monitoring stations?

7.3 Question 3: How can these air quality monitoring devices be applied for public benefit?

7.4 Limitations

All well and good doing all this, but if people don't have the ease of use (Need more cycle path) and education (on effects of pollution).

7.4.1 Public preferences

People care more about speed.

7.4.2 Timing of project

An analysis of seasonal and diurnal variation at a UK city is presented by Bigi and Harrison [2010] [REF]. NO2 concentrations in Europe tend to be higher in the winter than in the summer season. Hence, observations in the summer season had a lower chance to be detected as outliers by our method.

As noted in Section 5.2, Bath is surrounded by hills. This may lead to a layer of cloud forming above Bath, thereby trapping low quality air. Hasenfratz et al. (2015) found that this effect is prevalent in winter in Zurich, Switzerland. Unfortunately, this project is being conducted throughout the summer and so this effect may not be experienced.

Might not be as high levels since schools broken up for holiday.

Given the time restrictions of the project (June 2018 – September 2018), it was not possible to collect data over a sufficient time period to analyse sensor drift in the low-quality sensors.

7.4.3 Resource constraints

One of the main restrictions faced during this project was the availability of one rider and only one constructed air quality monitoring device. Consequently, a limited amount of data could be collected.

7.4.4 Equipment breakages

Broken ozone and humidity sensors half-way through data collection.

Flat tyres

Chapter 8

Conclusion

In this chapter, we conclude the project and provide recommendations for future work.

8.1 Conclusions

8.2 Future work

As a result of building this air quality monitoring device and conducting thorough testing, we identified a number of limitations, detailed in Section 7.4. These have helped us to narrow our focus and design improvements for future iterations of the device.

Firstly, we would re-design the case of the device to enclose all of the sensors in a miniature controlled environment. The case would have one, or potentially multiple, inlet fans that draw air into the container at a controlled – and measured – flow rate. This is so... In addition, the encased design would waterproof the device, making it much less prone to the rain-related issues we encountered in this project.

Secondly, we would like to replace the Shinyei PPD42 sensor with a slightly more expensive alternative. ...

MQTT Turning health-optimal routing into application based on real-time data Cycle-to-work scheme. This is how you would make it into a product... Event-based - if device has built-in fan, could just park bike outside ground and

let it collect every 10 seconds. Analyse how the health-optimal routing changes throughout the day by collecting data every hour across the whole city. Test laser-based particulate sensors instead. With fan.

- Higher quality sensors / different sensors
- Fully enclosed in case (waterproof and controlled air flow)
- Calibrate the ozone sensor using a controlled environment.
- Use focused NOx sensor next time.
- Add SIM to Raspberry Pi so does not rely on WiFi. Data cost is ridiculously low.
- Heatwave in the UK during data collection. No rain.
- Routing app

8.3 Additional application post-project completion

In collaboration with the University of Bath's Department of Mathematical Sciences, this air quality monitoring device is being tested in the capital of Mongolia, Ulaanbaatar. The research team are in the process of creating a project proposal and will use the device to demonstrate a physical application that can collect data for use in mathematical pollution modelling. The air quality in Ulaanbaatar is extremely poor, mainly due to households burning low-quality coal for heating in the cold climate.

Some adjustments were made to the device to enable this application. Firstly, the triangular bicycle bag was replaced with a plastic container and fans fitted into the walls of the container to induce air flow over the sensors. This makes the device relatively waterproof as the sensors are no longer directly exposed to the environment. Secondly, we replaced the MQ135 sensor with the dedicated NO_2 sensor that was recently released by MikroElektronika. This should enable accurate NO_2 measurements. Thirdly, we secured all sensors physically inside the container, thereby avoiding power disruption issues from vibrations and making the device more robust. Figure X shows the modified device.

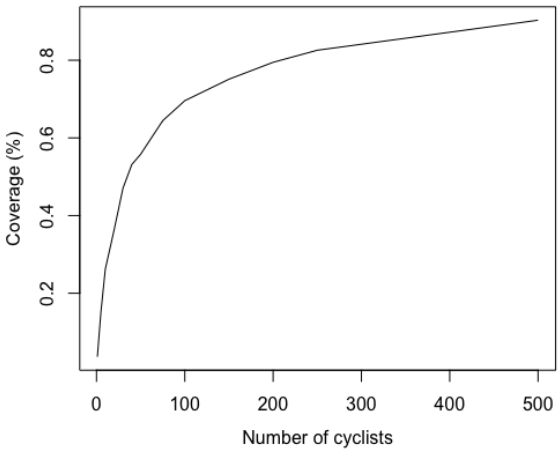


Figure 8-1: Percentage of Bath theoretically covered by n cyclists. Over 50% of the city can be covered with only 50 cyclists, with diminishing returns to scale as the number of cyclists increases.

Bibliography

Abraham, S. and Li, X., 2014. A cost-effective wireless sensor network system for indoor air quality monitoring applications. *Procedia Computer Science*, 34, pp.165–171.

Adams, M., DeLuca, P., Corr, D. and Kanaroglou, P., 2012. Mobile Air Monitoring: Measuring Change in Air Quality in the City of Hamilton, 2005-2010. *Social Indicators Research*, 108(2), pp.351–364.

Alkandari, A.A. and Moein, S., 2018. Implementation of Monitoring System for Air Quality using Raspberry Pi: Experimental Study. *Indonesian Journal of Electrical Engineering and Computer Science*, 10(1), pp.43–49.

Alvear, Ó., Zamora, W., Calafate, C.T., Cano, J.C. and Manzoni, P., 2016. EcoSensor: Monitoring environmental pollution using mobile sensors. *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks*, 21–24 June 2016 Coimbra. Washington: IEEE, pp.1–6.

Alvear, O., Zema, N.R., Natalizio, E. and Calafate, C.T., 2017. Using UAV-based systems to monitor air pollution in areas with poor accessibility. *Journal of Advanced Transportation*.

Andersen, A., Krøgholt, P., Bierre, S. and Tabard, A., 2012. *NoxDroid - A Bicycle Sensor System for Air Pollution Monitoring* [Online]. Copenhagen: IT University Copenhagen. Available from: `http://noxdroid.org/` [Accessed 18/07/2018].

Anderson, J.O., Thundiyil, J.G. and Stolbach, A., 2012. Clearing the Air: A

Review of the Effects of Particulate Matter Air Pollution on Human Health. *Journal of Medical Toxicology*, 8(2), pp.166–175.

Apte, J.S., Messier, K.P., Gani, S., Brauer, M., Kirchstetter, T.W., Lunden, M.M., Marshall, J.D., Portier, C.J., Vermeulen, R.C. and Hamburg, S.P., 2017. High-Resolution Air Pollution Mapping with Google Street View Cars: Exploiting Big Data. *Environmental Science and Technology*, 51(12), pp.6999–7008.

Balasubramaniyan, C. and Manivannan, D., 2016. IoT enabled Air Quality Monitoring System (AQMS) using Raspberry Pi. *Indian Journal of Science and Technology*, 9(39).

BANES, 2017. *Bath Air Quality Action Plan – Consultation Draft (final)*. Bath: Bath and North East Somerset Council.

Beelen, R., Raaschou-Nielsen, O., Stafoggia, M., Andersen, Z.J., Weinmayr, G., Hoffmann, B., Wolf, K., Samoli, E., Fischer, P., Nieuwenhuijsen, M., Vineis, P., Xun, W.W., Katsouyanni, K., Dimakopoulou, K., Oudin, A., Forsberg, B., Modig, L., Havulinna, A.S., Lanki, T., Turunen, A., Oftedal, B., Nystad, W., Nafstad, P., De Faire, U., Pedersen, N.L., Östenson, C.G., Fratiglioni, L., Penell, J., Korek, M., Pershagen, G., Eriksen, K.T., Overvad, K., Ellermann, T., Eeftens, M., Peeters, P.H., Meliefste, K., Wang, M., Bueno-De-Mesquita, B., Sugiri, D., Krämer, U., Heinrich, J., De Hoogh, K., Key, T., Peters, A., Hampel, R., Concin, H., Nagel, G., Ineichen, A., Schaffner, E., Probst-Hensch, N., Künzli, N., Schindler, C., Schikowski, T., Adam, M., Phuleria, H., Vilier, A., Clavel-Chapelon, F., Declercq, C., Grioni, S., Krogh, V., Tsai, M.Y., Ricceri, F., Sacerdote, C., Galassi, C., Migliore, E., Ranzi, A., Cesaroni, G., Badaloni, C., Forastiere, F., Tamayo, I., Amiano, P., Dorronsoro, M., Katsoulis, M., Trichopoulou, A., Brunekreef, B. and Hoek, G., 2014. Effects of long-term exposure to air pollution on natural-cause mortality: An analysis of 22 European cohorts within the multicentre ESCAPE project. *The Lancet*, 383(9919), pp.785–795.

Bigazzi, A.Y. and Figliozzi, M.A., 2014. Review of urban bicyclists' intake and uptake of traffic-related air pollution. *Transport Reviews*, 34(2), pp.221–245.

Boeing, G., 2017. OSMnx: New methods for acquiring, constructing, analyzing,

and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65, pp.126–139.

Briggs, D.J., Collins, S., Elliott, P., Fischer, P., Kingham, S., Lebret, E., Pryl, K., Van Reeuwijk, H., Smallbone, K. and Van Der Veen, A., 1997. Mapping urban air pollution using GIS: A regression-based approach. *International Journal of Geographical Information Science*, 11(7), pp.699–718.

Broach, J., Dill, J. and Gliebe, J., 2012. Where do cyclists ride? a route choice model developed with revealed preference gps data. *Transportation Research Part A: Policy and Practice*, 46(10), pp.1730 – 1740.

Brook, R., Franklin, B., Cascio, W., Hong, Y., Howard, G., Lipsett, M., Luepker, R., Mittleman, M., Samet, J., Smith Jr., S. and Tager, I., 2004. Air pollution and cardiovascular disease: A statement for healthcare professionals from the expert panel on population and prevention science of the American Heart Association. *Circulation*, 109(21), pp.2655–2671.

Castell, N., Dauge, F.R., Schneider, P., Vogt, M., Lerner, U., Fishbain, B., Broday, D. and Bartonova, A., 2017. Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates? *Environment International*, 99, pp.293–302.

Clements, A.L., Griswold, W.G., RS, A., Johnston, J.E., Herting, M.M., Thorson, J., Collier-Oxandale, A. and Hannigan, M., 2017. Low-Cost Air Quality Monitoring Tools: From Research to Practice (A Workshop Summary). *Sensors*, 17(11), p.2478.

Courthold, N., 07/08/2018. *Air quality monitoring in bath.* Email to E. Jeffery.

De Hartog, J.J., Boogaard, H., Nijland, H. and Hoek, G., 2010. Do the health benefits of cycling outweigh the risks? *Environmental health perspectives*, 118(8), p.1109.

DEFRA, 2017a. *Defra National Statistics Release: Air quality statistics in the UK 1987 to 2017.* London: DEFRA, (ENV02).

DEFRA, 2017b. *UK plan for tackling roadside nitrogen dioxide concentrations.* London: DEFRA.

DEFRA, 2018a. *Daily Air Quality Index* [Online]. London, UK: Department for Environment Food and Rural Affairs. Available from: `https://uk-air.defra.gov.uk/air-pollution/daqi` [Accessed 29/03/2018].

DEFRA, 2018b. *UK-AIR: Air Information Resource* [Online]. London: DEFRA. Available from: `https://uk-air.defra.gov.uk/` [Accessed 28/03/2018].

Desai, N.S. and Alex, J.S.R., 2017. IoT based air pollution monitoring and predictor system on Beagle bone black. In: S.R.S. Prabaharan, ed. *2017 International Conference On Nextgen Electronic Technologies: Silicon to Software*, 23–25 March 2017 Chennai. Washington: IEEE, pp.367–370.

Devarakonda, S., Sevusu, P., Liu, H., Liu, R., Iftode, L. and Nath, B., 2013. Real-time air quality monitoring through mobile sensing in metropolitan areas. In: R.L. Grossman and R. Uthurusamy, eds. *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 11–14 August 2013 Chicago. New York: ACM, pp.15–23.

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1), pp.269–271.

Dockery, D., Pope, C., xu, X., Spengler, J., Ware, J., Fay, M., Ferris, B. and Speizer, F., 1993. An Association between Air Pollution and Mortality in Six U.S. Cities. *New England Journal of Medicine*, 329(24), pp.1753–1759.

Ebi, K.L. and McGregor, G., 2008. Climate change, tropospheric ozone and particulate matter, and health impacts. *Environmental Health Perspectives*, 116(11), pp.1449–1455.

Elen, B., Peters, J., Poppel, M. van, Bleux, N., Theunis, J., Reggente, M. and Standaert, A., 2013. The Aeroflex: A bicycle for mobile air quality measurements. *Sensors*, 13(1), pp.221–240.

European Environment Agency, 2018. *European Air Quality Index: current air quality information at your finger tips* [Online]. Copenhagen, Denmark: EEA. Available from: `https://www.eea.europa.eu/highlights/european-air-quality-index-current` [Accessed 29/03/2018].

European Parliament, 2008. Directive 2008/50/ec of the European Parliament and of the Council of 21st May 2008 on ambient air quality and cleaner air for Europe [2008] *OJ* L151/1.

Faustini, A., Rapp, R. and Forastiere, F., 2014. Nitrogen dioxide and mortality: Review and meta-analysis of long-term studies. *European Respiratory Journal*, 44(3), pp.744–753.

Ferdoush, S. and Li, X., 2014. Wireless sensor network system design using Raspberry Pi and Arduino for environmental monitoring applications. *Procedia Computer Science*, 34, pp.103–110.

Fuertes, W., Carrera, D., Villacis, C., Toulkeridis, T., Galarraga, F., Torres, E. and Aules, H., 2016. Distributed system as internet of things for a new low-cost, air pollution wireless monitoring on real time. In: S.J. Turner and Y. Yao, eds. *Proceedings - 2015 IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications, DS-RT 2015*, 14–16 October 2015 Chengdu. New York: ACM, pp.58–67.

Garnier, O., 2017. *Affordable air quality monitoring ? myth or reality?* Dissertation (B.Sc.). University of Bath, Bath.

Google, 2017a. *Environmental Report: 2017 progress update.* Mountain View, CA: Google.

Google, 2017b. *Maps Javascript API: Heatmaps* [Online]. Mountain View, CA: Google. Available from: `https://developers.google.com/maps/documentation/javascript/examples/layer-heatmap` [Accessed 18/02/2018].

Guarnieri, M. and Balmes, J.R., 2014. Outdoor air pollution and asthma. *The Lancet*, 383(9928), pp.1581–1592.

Hagemann, R., Corsmeier, U., Kottmeier, C., Rinke, R., Wieser, A. and Vogel, B., 2014. Spatial variability of particle number concentrations and NOx in the Karlsruhe (Germany) area obtained with the mobile laboratory 'AERO-TRAM'. *Atmospheric Environment*, 94, pp.341–352.

Hagler, G., Thoma, E. and Baldauf, R., 2010. High-resolution mobile monitoring of carbon monoxide and ultrafine particle concentrations in a near-road environment. *Journal of the Air and Waste Management Association*, 60(3), pp.328–336.

Hasenfratz, D., Saukh, O., Walser, C., Hueglin, C., Fierz, M., Arn, T., Beutel, J. and Thiele, L., 2015. Deriving high-resolution urban air pollution maps using mobile sensor nodes. *Pervasive and Mobile Computing*, 16, pp.268–285.

Hoang, D., Strufe, T., Le, Q., Bui, P., Pham, T., Thai, N., Le, T. and Schweizer, I., 2013. Processing and visualizing traffic pollution data in Hanoi City from a wireless sensor network. In: D. Turgut and N. Aschenbruck, eds. *38th Annual IEEE Conference on Local Computer Networks - Workshops*, 21–24 October 2013 Sydney. Washington: IEEE, pp.48–55.

Hoek, G., Beelen, R., Hoogh, K. de, Vienneau, D., Gulliver, J., Fischer, P. and Briggs, D., 2008. A review of land-use regression models to assess spatial variation of outdoor air pollution. *Atmospheric Environment*, 42(33), pp.7561–7578.

Hui, M., 2011. *Energy harvesting bicycle*. Dissertation (B.Sc.). California Polytechnic State University, California.

Ibrahim, M., Elgamri, A., Babiker, S. and Mohamed, A., 2015. Internet of Things based Smart Environmental Monitoring using the Raspberry-Pi Computer. *The Fifth International Conference on Digital Information Processing and Communications*, 7–9 October 2015 Sierre. IEEE, Sierre, Switzerland: IEEE, pp.159–164.

Jabbar, A.A., Aicardi, I., Grasso, N. and Piras, M., 2017. Urban data collection using a bike mobile system with a foss architecture. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 42(4), pp.3–9.

Jerrett, M., Arain, A., Kanaroglou, P., Beckerman, B., Potoglou, D., Sahsuvaroglu, T., Morrison, J. and Giovis, C., 2005. A review and evaluation of intraurban air pollution exposure models. *Journal of Exposure Analysis and Environmental Epidemiology*, 15(2), pp.185–204.

Jerrett, M., Shankardass, K., Berhane, K., Gauderman, W.J., Künzli, N., Avol, E., Gilliland, F., Lurmann, F., Molitor, J.N., Molitor, J.T. et al., 2008. Traffic-related air pollution and asthma onset in children: a prospective cohort study with individual exposure measurement. *Environmental Health Perspectives*, 116(10), p.1433.

Kanaroglou, P.S., Jerrett, M., Morrison, J., Beckerman, B., Arain, M.A., Gilbert, N.L. and Brook, J.R., 2005. Establishing an air pollution monitoring network for intra-urban population exposure assessment: A location-allocation approach. *Atmospheric Environment*, 39(13), pp.2399–2409.

Kuhlbusch, T.A., Quincey, P., Fuller, G.W., Kelly, F., Mudway, I., Viana, M., Querol, X., Alastuey, A., Katsouyanni, K., Weijers, E., Borowiak, A., Gehrig, R., Hueglin, C., Bruckmann, P., Favez, O., Sciare, J., Hoffmann, B., EspenYttri, K., Torseth, K., Sager, U., Asbach, C. and Quass, U., 2014. New Directions: The future of European urban air quality monitoring. *Atmospheric Environment*, 87, pp.258–260.

Lee, S., Jo, J., Kim, Y. and Stephen, H., 2014. A Framework for Environmental Monitoring with Arduino-Based Sensors Using Restful Web Service. *2014 IEEE International Conference on Services Computing*, 27 June–2 July 2014 Anchorage. Washington: IEEE, pp.275–282.

Min, K.T., Forys, A. and Schmid, T., 2014. Demonstration abstract: AirFeed - Indoor real time interactive air quality monitoring system. In: A. Wolisz, ed. *IPSN 2014 - Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, 15–17 April 2014 Berlin. Washington: IEEE, pp.325–326.

Office for National Statistics, 2011. *2011 Census aggregate data* [Online]. London: Office for National Statistics. Available from: `https://www.ons.gov.uk/census/2011census` [Accessed 19/07/2018].

Peters, J., Theunis, J., Van Poppel, M. and Berghmans, P., 2013. Monitoring pm10 and ultrafine particles in urban environments using mobile measurements. *Aerosol and Air Quality Research*, 13(2), pp.509–522.

Piedrahita, R., Xiang, Y., Masson, N., Ortega, J., Collier, A., Jiang, Y., Li, K., Dick, R.P., Lv, Q., Hannigan, M. and Shang, L., 2014. The next generation of low-cost personal air quality sensors for quantitative exposure monitoring. *Atmospheric Measurement Techniques*, 7(10), pp.3325–3336.

Pope, C.A., Burnett, R.T., Thun, M.J., Calle, E.E., Krewski, D., Ito, K. and Thurston, G.D., 2002. Lung cancer, cardiopulmonary mortality, and long-term exposure to fine particulate air pollution. *JAMA*, 287(9), pp.1132–1141.

Pope, C.A., Thun, M.J., Namboodiri, M.M., Dockery, D.W., Evans, J.S., Speizer, F.E., Heath, C.W. et al., 1995. Particulate air pollution as a predictor of mortality in a prospective study of us adults. *American Journal of Respiratory and Critical Care Medicine*, 151(3), pp.669–674.

Rahman, M., Rahman, A., Afrin, A., Hong, H.J., Tsai, P.H., Uddin, M., Venkatasubramanian, N. and Hsu, C.H., 2017. Adaptive sensing using internet-of-things with constrained communications. In: A. Dubey and M. Garca-Valls, eds. *Proceedings of the 16th Workshop on Adaptive and Reflective Middleware, ARM 2017*, 11–15 December 2017 Las Vegas. New York: ACM, pp.6–12.

Ramos, C.A., Wolterbeek, H.T. and Almeida, S.M., 2016. Air pollutant exposure and inhaled dose during urban commuting: a comparison between cycling and motorized modes. *Air Quality, Atmosphere & Health*, 9(8), pp.867–879.

Re, G.L., Peri, D. and Vassallo, S.D., 2014. Urban Air Quality Monitoring Using Vehicular Sensor Networks. In: S. Gaglio and G.L. Re, eds. *Advances onto the Internet of Things*, New York, US: Springer, vol. 260, pp.311–323.

Rousseeuw, P.J. and Croux, C., 1993. Alternatives to the median absolute deviation. *Journal of the American Statistical association*, 88(424), pp.1273–1283.

Schepers, P., Fishman, E., Beelen, R., Heinen, E., Wijnen, W. and Parkin, J., 2015. The mortality impact of bicycle paths and lanes related to physical activity, air pollution exposure and road safety. *Journal of Transport & Health*, 2(4), pp.460–473.

Sharker, M.H. and Karimi, H.A., 2014. Computing least air pollution exposure routes. *International Journal of Geographical Information Science*, 28(2), pp.343–362.

Skov-Petersen, H., Barkow, B., Lundhede, T. and Jacobsen, J.B., 2018. How do cyclists make their way? - a gps-based revealed preference study in copenhagen. *International Journal of Geographical Information Science*, 32(7), pp.1469–1484.

Smallbone, K., 2012. *Individuals interpretation of air quality information: customer insight and awareness study*. Brighton: University of Brighton, (719).

Spinelle, L., Gerboles, M., Villani, M., Aleixandre, M. and Bonavitacola, F., 2015. Field calibration of a cluster of low-cost available sensors for air quality monitoring. Part A: Ozone and nitrogen dioxide. *Sensors and Actuators, B: Chemical*, 215, pp.249–257.

Spinelle, L., Gerboles, M., Villani, M.G., Aleixandre, M. and Bonavitacola, F., 2017. Field calibration of a cluster of low-cost commercially available sensors for air quality monitoring. Part B: NO, CO and CO2. *Sensors and Actuators, B: Chemical*, 238, pp.706–715.

Sun, L., Wong, K.C., Wei, P., Ye, S., Huang, H., Yang, F., Westerdahl, D., Louie, P.K., Luk, C.W. and Ning, Z., 2016. Development and application of a next generation air sensor network for the Hong Kong marathon 2015 air quality monitoring. *Sensors*, 16(2), p.211.

Thorpe, M.R., 2017. *Improving low cost sensing with ad-hoc mesh networks and machine learning*. Thesis (M.Sc.). University of Bath, Bath.

Tukey, J.W., 1977. *Exploratory data analysis*. 2nd ed. Boston: Addison-Wesley.

Turner, M.C., Jerrett, M., Pope, C.A., Krewski, D., Gapstur, S.M., Diver, W.R., Beckerman, B.S., Marshall, J.D., Su, J., Crouse, D.L. and Burnett, R.T., 2016. Long-Term Ozone Exposure and Mortality in a Large Prospective Study. *American Journal of Respiratory and Critical Care Medicine*, 193(10), pp.1134–1142.

Uva, M., Falcone, R., McClellan, A. and Ostapowicz, E., 2009. *Preliminary Screening System for Ambient Air Quality in Southeast Philadelphia.* Philadelphia: Drexel University, (ECE-19).

Wallace, J., Corr, D., Deluca, P., Kanaroglou, P. and McCarry, B., 2009. Mobile monitoring of air pollution in cities: The case of Hamilton, Ontario, Canada. *Journal of Environmental Monitoring*, 11(5), pp.998–1003.

Weijers, E., Khlystov, A., Kos, G. and Erisman, J., 2004. Variability of particulate matter concentrations along roads and motorways determined by a moving measurement unit. *Atmospheric Environment*, 38(19), pp.2993–3002.

Westerdahl, D., Fruin, S., Sax, T., Fine, P. and Sioutas, C., 2005. Mobile platform measurements of ultrafine particles and associated pollutant concentrations on freeways and residential streets in Los Angeles. *Atmospheric Environment*, 39(20), pp.3597–3610.

WHO, 2002. *Monitoring ambient air quality for health impact assessment.* Geneva: World Health Organisation.

WHO, 2016. *Ambient air pollution: A global assessment of exposure and burden of disease.* Geneva: World Health Organisation.

Wong, K.J., Chua, C.C. and Li, Q., 2009. Environmental Monitoring Using Wireless Vehicular Sensor Networks. In: Y. Zhang, Z. Niu and L. Cuthbert, eds. *5th International Conference on Wireless Communications, Networking and Mobile Computing*, 24–26 September 2009 Beijing. Washington: IEEE, pp.1–4.

Yang, Y., Yeo, J. and Priya, S., 2012. Harvesting energy from the counterbalancing (Weaving) movement in bicycle riding. *Sensors*, 12(8), pp.10248–10258.

Zimmerman, N., Presto, A.A., Kumar, S.P., Gu, J., Hauryliuk, A., Robinson, E.S., Robinson, A.L. and Subramanian, R., 2018. A machine learning calibration model using random forests to improve sensor performance for lower-cost air quality monitoring. *Atmospheric Measurement Techniques*, 11(1), p.291.

Zoest, V.M. van, Stein, A. and Hoek, G., 2018. Outlier detection in urban air quality sensor networks. *Water, Air, & Soil Pollution*, 229(4), p.111.

Appendix A

Installation instructions

A.1 GPS

To get the USB GPS module to work with the Raspberry Pi, a number of libraries must first be installed.

```
sudo apt-get install gpsd gpsd-clients python-gps ntp
```

Reboot the RPi. Next, the `/etc/default/gpsd` file needs to be edited.

```
START_DAEMON="true"
USBAUTO="true"
DEVICES="/dev/ttyUSB0"
GPSD_OPTIONS="-F /var/run/gpsd.sock -b -n"
```

The GPSD service now needs to be restarted.

```
sudo systemctl enable gpsd
sudo systemctl start gpsd
sudo systemctl status gpsd
```

Now run `cgps -s` to test whether the GPS is connecting to satellites. The longitude and latitude fields should be populated.

A.2 NTP

To enable the RPi to use the GPS as an NTP server, the `/etc/ntp.conf` file needs to be edited. Add the following lines underneath the `server 3.debian.pool.ntp.org iburst` line.

```
# gps ntp
server 127.127.28.0 minpoll 4
fudge 127.127.28.0  time1 0.183 refid NMEA
server 127.127.28.1 minpoll 4 prefer
fudge 127.127.28.1  time1 0.183 refid PPS
```

There should also be a line starting with `restrict`. Edit this line to be:

```
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

The NTP service now needs to be restarted.

```
sudo systemctl enable ntp
sudo systemctl start ntp
sudo systemctl status ntp
```

Now run `ntpq -p` to test whether the GPS is working as an NTP server. It should display time from `SHM(0)` and `SHM(1)`.

Appendix B

Code

B.1 Main RPi program

```
#!/usr/bin/env python

from __future__ import print_function
import spidev
import time
import math
import gpsd
import requests
import os
import pigpio
import threading
import RPi.GPIO as GPIO
import sys
import DHT22
import atexit
import Adafruit_DHT

class MCP3204:
        def __init__(self, spi_channel=0):
                self.spi_channel = spi_channel
                self.conn = spidev.SpiDev(1,spi_channel)
                self.conn.max_speed_hz = 50000 # 1MHz
```

```
        def __del__( self ):
                self.close

        def close(self):
                if self.conn != None:
                        self.conn.close
                        self.conn = None

        def bitstring(self, n):
                s = bin(n)[2:]
                return '0'*(8-len(s)) + s

        def read(self, adc_channel=0):
                # build command
                cmd  = 128 # start bit
                cmd +=  64 # single end / diff
                if adc_channel % 2 == 1:
                        cmd += 8
                if (adc_channel/2) % 2 == 1:
                        cmd += 16
                if (adc_channel/4) % 2 == 1:
                        cmd += 32

                # send & receive data
                reply_bytes = self.conn.xfer2([cmd, 0, 0, 0])

                reply_bitstring = ''.join(self.bitstring(n)
                   ↪ for n in reply_bytes)
                # print reply_bitstring

                # see also... http://akizukidenshi.com/
                   ↪ download/MCP3204.pdf (page.20)
                reply = reply_bitstring[5:19]
                return int(reply, 2)

class MCP3551:
        def __init__(self, spi_channel=0):
```

```
            self.spi_channel = spi_channel
            self.conn = spidev.SpiDev(0,spi_channel)
            self.conn.mode = 0b11
            self.conn.max_speed_hz = 50000 # 1MHz

    def __del__( self ):
            self.close

    def close(self):
            if self.conn != None:
                    self.conn.close
                    self.conn = None

    def bitstring(self, n):
            s = bin(n)[2:]
            return '0'*(8-len(s)) + s

    def read(self, adc_channel=0):
            bytes = self.conn.readbytes(3)
            return bytes

    def convert(self, bs):

        ADC_Byte3 = bs[0]
        ADC_Byte2 = bs[1]
        ADC_Byte1 = bs[2]

        # store 24 bits in ADC_Value
        ADC_Value = ADC_Byte3;
        ADC_Value = (ADC_Value << 8) | ADC_Byte2;
        ADC_Value = (ADC_Value << 8) | ADC_Byte1;

        # OVH (false) condition
        if (ADC_Byte3 == 0x60 and ADC_Byte1 > 0):
            ADC_Value = 3000000             # Vin+ >=
               ↪ Vref ; Vin- = GND

        # OVL (false) condition
```

```
            elif (ADC_Byte3 == 0x9F and ADC_Byte1 > 0):
                ADC_Value = -3000000            # Vin- >=
                    ↪ Vref ; Vin+ = GND

            else:
                if ((ADC_Byte3 & 0x20) >> 5):      # Bit 21 =
                    ↪  1 ; value is negative
                     ADC_Value = ADC_Value - 4194304    # for
                        ↪ 22 bit resolution

            return ADC_Value

class sensor:
   """
   A class to read a Shinyei PPD42NS Dust Sensor, e.g. as
      ↪ used
   in the Grove dust sensor.
   This code calculates the percentage of low pulse time and
   calibrated concentration in particles per 1/100th of a
      ↪ cubic
   foot at user chosen intervals.
   You need to use a voltage divider to cut the sensor output
   voltage to a Pi safe 3.3V (alternatively use an in-line
   20k resistor to limit the current at your own risk).
   """

   def __init__(self, pi, gpio):
      """
      Instantiate with the Pi and gpio to which the sensor
      is connected.
      """

      self.pi = pi
      self.gpio = gpio

      self._start_tick = None
      self._last_tick = None
      self._low_ticks = 0
```

```
      self._high_ticks = 0

      pi.set_mode(gpio, pigpio.INPUT)

      self._cb = pi.callback(gpio, pigpio.EITHER_EDGE, self.
         ↪ _cbf)

   def read(self):
      """
      Calculates the percentage low pulse time and calibrated
      concentration in particles per 1/100th of a cubic foot
      since the last read.
      For proper calibration readings should be made over
      30 second intervals.
      Returns a tuple of gpio, percentage, and concentration.
      """
      interval = self._low_ticks + self._high_ticks

      if interval > 0:
         ratio = float(self._low_ticks)/float(interval)*100.0
         conc = 1.1*pow(ratio,3)-3.8*pow(ratio,2)+520*ratio
            ↪ +0.62;
      else:
         ratio = 0
         conc = 0.0

      self._start_tick = None
      self._last_tick = None
      self._low_ticks = 0
      self._high_ticks = 0

      return (self.gpio, ratio, conc)

   def _cbf(self, gpio, level, tick):

      if self._start_tick is not None:

         ticks = pigpio.tickDiff(self._last_tick, tick)
```

```
            self._last_tick = tick

            if level == 0: # Falling edge.
               self._high_ticks = self._high_ticks + ticks
               #print("Falling edge")

            elif level == 1: # Rising edge.
               self._low_ticks = self._low_ticks + ticks
               #print("Rising edge")

            else: # timeout level, not used
               #print("Timeout level")
               pass

         else:
            self._start_tick = tick
            self._last_tick = tick


      def pcs_to_ugm3(self, concentration_pcf):
         '''
         Convert concentration of PM2.5 particles per 0.01
            ↪ cubic feet to ug/ metre cubed
         this method outlined by Drexel University students
            ↪ (2009) and is an approximation
         does not contain correction factors for humidity and
            ↪ rain
         '''

         if concentration_pcf < 0:
            raise ValueError('Concentration cannot be a
               ↪ negative number')

         # Assume all particles are spherical, with a density
            ↪ of 1.65E12 ug/m3
         densitypm25 = 1.65 * math.pow(10, 12)
```

```
        # Assume the radius of a particle in the PM2.5
            ↪ channel is .44 um
        rpm25 = 0.44 * math.pow(10, -6)

        # Volume of a sphere = 4/3 * pi * radius^3
        volpm25 = (4/3) * math.pi * (rpm25**3)

        # mass = density * volume
        masspm25 = densitypm25 * volpm25

        # parts/m3 =  parts/foot3 * 3531.5
        # ug/m3 = parts/m3 * mass in ug
        concentration_ugm3 = concentration_pcf * 3531.5 *
            ↪ masspm25

        return concentration_ugm3


    def ugm3_to_aqi(self, ugm3):
        '''
        Convert concentration of PM2.5 particles in ug/ metre
            ↪  cubed to the USA
        Environment Agency Air Quality Index - AQI
        https://en.wikipedia.org/wiki/Air_quality_index
        Computing_the_AQI
        https://github.com/intel-iot-devkit/upm/pull/409/
            ↪ commits/
            ↪ ad31559281bb5522511b26309a1ee73cd1fe208a?diff=
            ↪ split
        '''

        cbreakpointspm25 = [ [0.0, 12, 0, 50],\
                        [12.1, 35.4, 51, 100],\
                        [35.5, 55.4, 101, 150],\
                        [55.5, 150.4, 151, 200],\
                        [150.5, 250.4, 201, 300],\
                        [250.5, 350.4, 301, 400],\
                        [350.5, 500.4, 401, 500], ]
```

```
        C=ugm3

        if C > 500.4:
            aqi=500

        else:
           for breakpoint in cbreakpointspm25:
               if breakpoint[0] <= C <= breakpoint[1]:
                   Clow = breakpoint[0]
                   Chigh = breakpoint[1]
                   Ilow = breakpoint[2]
                   Ihigh = breakpoint[3]
                   aqi=(((Ihigh-Ilow)/(Chigh-Clow))*(C-Clow))
                      ↪ +Ilow

        return aqi


def getSerial():
    # Extract serial from cpuinfo file
    cpu_serial = "0000000000000000"
    try:
        f = open('/proc/cpuinfo', 'r')
        for line in f:
            if line[0:6]=='Serial':
                cpu_serial = line[10:26]
        f.close()
    except:
        cpu_serial = "ERROR00000000000"

    return cpu_serial


class GasesThread(threading.Thread):

    def __init__(self):
        threading.Thread.__init__(self)
```

```
        self.stopped = False
        self.mq131 = 0
        self.mq135 = 0

    def run(self):
        print("Gases thread running")

        spi1 = MCP3551()
        spi2 = MCP3204(2)

        while not self.stopped:
            mq_131_adc_value = spi1.read(0)
            mq_135_adc_value = spi2.read(1)

            if mq_131_adc_value[0] != 255:
                mq_131_adc_value = spi1.convert(
                    ↪ mq_131_adc_value)
                #### NEED TO CHECK THIS ####
                mq131_voltage = (float(mq_131_adc_value) / (
                    ↪ math.pow(2, 21) - 1 / 2)) * 5.0
                #print("O3: %.3f" % mq131_voltage)
                self.mq131 = mq131_voltage

            mq135_voltage = (float(mq_135_adc_value) / 4096)
                ↪ * 5.0
            #print("Gases: %.3f" % mq135_voltage)
            self.mq135 = mq135_voltage

            time.sleep(1)


class ParticulatesThread(threading.Thread):

    def __init__(self):
        threading.Thread.__init__(self)
        self.stopped = False
        self.pm = 0
        self.temp = 0
```

```
        self.humidity = 0

    def run(self):
        print("Particulates thread running")

        pi = pigpio.pi() # Connect to Pi.
        s = sensor(pi, 23) # set the GPIO pin number

        dht22 = DHT22.sensor(pi, 22) # DHT22: set the GPIO
           ↪ pin number

        while not self.stopped:
            try:
                # Sleep for 10 seconds (minus code execution
                   ↪ time)
                time.sleep(7.5)

                # Get the gpio, ratio and concentration in
                   ↪ particles / 0.01 ft3
                g, ratio, c = s.read()

                # Get new relative humidity and temperature
                   ↪ reading.
                dht22.trigger()
                time.sleep(0.5)
                temp = dht22.temperature()
                humidity = dht22.humidity()
                #humidity, temp = Adafruit_DHT.read_retry(22,
                   ↪  22)

                if (c == 1114000.62):
                    c = 0
                    continue

                # Convert to SI units
                concentration_ugm3 = s.pcs_to_ugm3(c)

                # Store latest readings
```

```
                self.pm = concentration_ugm3
                if temp != None:
                    self.temp = temp
                if humidity != None:
                    self.humidity = humidity
            except Exception as e:
                print(e)
                # If error occurs, continue to avoid
                   ↪ measurement issues.
                continue

        pi.stop() # Disconnect from Pi.


if __name__ == '__main__':

    os.system("sudo pigpiod")

    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    led = 25
    GPIO.setup(led, GPIO.OUT)

    url = 'http://people.bath.ac.uk/ej301/route_api.php'

    try:
        # Get to GPS daemon
        gpsd.connect()

        # Start threads
        t1 = GasesThread()
        t2 = ParticulatesThread()
        t1.start()
        t2.start()

        stopped = False

        # Wait for GPS fix
```

```
        fixed = False
        while not fixed:
            try:
                p = gpsd.get_current()
                print("Waiting for connection...")
                if p.mode == 3:
                    # Modes: {1: No fix, 2: 2D fix, 3: 3D fix
                       ↪ }
                    # Need 3D fix for altitude
                    fixed = True
                    print("Connection obtained.")
                else:
                    time.sleep(5)
            except:
                continue

        # Sleep for 20s to allow initial readings to populate
        print("Sleeping for 15 seconds...")
        time.sleep(15)

        # Run main loop
        while not stopped:
            print("Running\n")
            count = 0

            #Get the datetime for the filename
            t0 = time.strftime("%Y-%m-%d_%H-%M-%S", time.
               ↪ gmtime())

            #Create the csv file for storing the values
            location = '/home/pi/Documents/data/route_data_'
               ↪ + t0 + '.csv'
            sent_location = '/home/pi/Documents/sent/
               ↪ route_data_' + t0 + '.csv'
            unsent_location = '/home/pi/Documents/unsent/
               ↪ route_data_' + t0 + '.csv'

            while count < 6:
```

```
# Get readings from threads
pm = t2.pm
mq131 = t1.mq131
mq135 = t1.mq135
temp = t2.temp
humidity = t2.humidity

# If on 10 second interval, print, save and
   ↪ send.
if int(time.time()) % 10 == 0:
    # Print readings
    print("PM: \t%.3fug^3" % pm)
    print("O3: \t%.3fV" % mq131)
    print("Gases: \t%.3fV" % mq135)
    print("Temp: \t%.3f" % temp)
    print("Humidity: \t%.3f" % humidity)

    # Get extra info
    serial = getSerial()
    unix_time = int(time.time()) - 1500
    packet = gpsd.get_current()
    latitude = packet.lat
    longitude = packet.lon
    altitude = packet.alt
    horizontal_speed = packet.hspeed

    # Switch on LED
    GPIO.output(led, 1)
    time.sleep(1)
    # Switch off
    GPIO.output(led, 0)

    # Concatenate data into one CSV string
    str_list = [serial, unix_time, latitude,
       ↪ longitude, pm, mq131, mq135,
       ↪ altitude, horizontal_speed, temp,
       ↪ humidity]
    str_list = [str(i) for i in str_list]
```

```
            str_list = ','.join(str_list)

            # Save data to file
            with open(location, 'a') as file:
               file.write(str_list)
               file.write('\n')
               file.close()
               print("Saved")

            print("")
            count += 1

            time.sleep(1) # Skip over the next second
               ↪  to avoid saving multiple times in
               ↪ same second

        # Check for error (all zeroes) in
           ↪ ParticulatesThread
        # If error, stop and restart the thread
        if pm == 0 or pm > 500:
            GPIO.output(led, 1)
            time.sleep(0.5)
            GPIO.output(led, 0)
            time.sleep(0.5)
            GPIO.output(led, 1)
            time.sleep(0.5)
            GPIO.output(led, 0)

        if temp == 0 or humidity == 0 or mq131 == 0:
            GPIO.output(led, 1)
            time.sleep(0.25)
            GPIO.output(led, 0)
            time.sleep(0.25)
            GPIO.output(led, 1)
            time.sleep(0.25)
            GPIO.output(led, 0)
```

```
        # Short sleep period to avoid excessive
            ↪ looping
        time.sleep(0.5)

    # Try to send file to database
    try:
        # Send file via http post
        files = {'file': open(location, 'rb')}
        r = requests.post(url, files=files)
        print(r.status_code, r.reason)
        print(r.text)

        # Try to also send any unsent files
        directory = '/home/pi/Documents/unsent/'
        sent_directory = '/home/pi/Documents/sent/'
        if len(os.listdir(directory)) > 0:
            for filename in os.listdir(directory):
                location_ = directory + filename
                sent_location_ = sent_directory +
                    ↪ filename
                files = {'file': open(location_, 'rb
                    ↪ ')}
                r = requests.post(url, files=files)
                print(r.status_code, r.reason)
                os.rename(location_, sent_location_)

        # Move file to sent folder
        os.rename(location, sent_location)

    except requests.ConnectionError:
        print("There is a network problem")
        ###################################
        # Need a re-send protocol here. #
        ###################################

        # Move file to unsent folder
        os.rename(location, unsent_location)
```

```
        # Stop threads when while loop ends
        t1.stopped = True
        t2.stopped = True

    except KeyboardInterrupt:
        print("Exception!")
        stopped = True
        t1.stopped = True
        t2.stopped = True
        sys.exit()
```

B.2 PHP API

```
<html>
<body>
<h1>Inserting route air quality data</h1>
<?php
require_once "DB.php";
$host="mysql5host.bath.ac.uk";
$user="ej301";
$password="Aesoh8oWPu1xohMu";
$database="ej301";
$dsn="mysql://$user:$password@$host/$database";
$db = DB::connect($dsn);
if (DB::isError($db)) {
die ($db->getMessage());
}

echo $_FILES['file']['size'];
var_dump($_POST);

$uploaddir = "";
$uploadfile = $uploaddir . basename( $_FILES['file']['name'])
   ↪ ;
echo $uploadfile;
echo "<br>";
```

```
if(move_uploaded_file($_FILES['file']['tmp_name'],
   ↪ $uploadfile))
{
  echo "The file has been uploaded successfully";
  echo "<br>";
}
else
{
  echo "There was an error uploading the file";
  echo "<br>";
}

//$fileContent = file_get_contents($uploadfile);
//echo $fileContent;
//echo "<br>";
//echo "<br>";

$data_upload_success = False;

$serial_num = 0;
$time = 0;
$rows_inserted = -1; // -1 as counter iterates one more than
   ↪ necessary
$file_size = filesize($uploadfile);
$csv_string = "";

$CSVfp = fopen($uploadfile, "r");
$CSVfp2 = fopen($uploadfile, "r");
if($CSVfp !== FALSE) {
 while(! feof($CSVfp)) {
  $csv_string .= fgets($CSVfp2);
  $data = fgetcsv($CSVfp, 1000, ",");
  print_r($data);
  echo "<br>";
  $rows_inserted++;
  if(is_array($data)) {
    $DataArr = array();
```

```
    $serial_num = $data[0];
    $time = $data[1];
    // longitude = 2
    // latitude = 3
    // pm_raw = 4
    // mq131 = 5
    // mq135 = 6
    // altitude = 7
    // horizontal_speed = 8
    // temp = 9
    // humidity = 10
    $DataArr[] = "('$data[0]', '$data[1]', '$data[2]', '$data
       ↪ [3]', '$data[4]', '$data[5]', '$data[6]', '$data
       ↪ [7]', '$data[8]', '$data[9]', '$data[10]')";
    //echo implode(',', $DataArr);
    //echo "<br>";
    //echo "<br>";

    $query = "INSERT INTO route_air_quality (serial_num, time
       ↪ , latitude, longitude, pm_raw, mq131, mq135,
       ↪ altitude, horizontal_speed, temp, humidity) VALUES
       ↪ ";
    $query .= implode(',', $DataArr);
    //echo $query;

    $result = $db->query($query);
    if (DB::isError($result)) {
    echo "<p>Not successful: " . $result->getMessage() .
                "</p>";
    }
    else {
      echo "<p>Done</p>\n";
      $data_upload_success = $data_upload_success;
    }
  }
}
if ($data_upload_success == True) {
  // Store monitoring data
```

```
   $SystemData = array();
   $SystemData[] = "('$serial_num', '$time', '$rows_inserted
      ↪ ', '$file_size', '$csv_string')";
   $monitoring_query = "INSERT INTO system_monitoring (
      ↪ serial_num, time, rows_inserted, file_size, csv)
      ↪ VALUES ";
   $monitoring_query .= implode(',', $SystemData);
   $result = $db->query($monitoring_query);
   if (DB::isError($result)) {
     echo "<p>System monitoring upload not successful: " .
        ↪ $result->getMessage() .
         "</p>";
   }
 }
}
fclose($CSVfp);

$db->disconnect();
?>
<p>Finished</p>
</body>
</html>
```

Appendix C

Data analysis code

C.1 Data structuring

C.2 Learning