IoT: Systems, Security and the Cloud Report

Jaka Mohorko
j.mohorko@ed.ac.uk
The University of Edinburgh

ABSTRACT

Air pollution is a growing concern in as the world population and urbanization increases. With the aim on providing more fine-grained data than the currently available stationary air quality monitors, we created an affordable and portable IoT air quality monitor prototype. The prototype uses several different sensors to measure pollutant levels relevant to monitoring both indoors and outdoors air quality. We make use of the Google Cloud platform for data storage and visualization and an Android device to act as a relay between the sensors and the Cloud.

1 INTRODUCTION

With growing amounts of urbanization, increases in world population, increased vehicle use and industrialization, air pollution is one of the major problems the world is facing. Gas emissions present a substantial risk to public health, affecting the well-being of those exposed. According to the World Health Organization (WHO), air pollution is estimated to be the cause of death for approximately seven million people annually worldwide. 9 out of 10 people are breathing air with high pollutant levels on a daily basis, which can often be the cause of various diseases. WHO reports that more than 80% of urban population lives in areas exceeding recommended air quality standards[12].

A common method of reporting daily air quality is the Air Quality Index (AQI). It is the measurement reported by most stationary air quality monitoring stations and is widely adopted as the most commonly used method of reporting the outdoor air quality. It uses the highest AQI value calculated from the amounts of Ozone, Carbon Monoxide, Sulfur Dioxide, Particulate Matter and Nitrogen Dioxide to classify the outdoor air into a scale ranging from "good" to "hazardous"[3].

When observing indoor air quality (IAQ), other gasses are often taken into consideration as well. In addition to the before mentioned gasses, Carbon Dioxide and Total Volatile Organic Compounds (TVOC) are of importance[1]. While air temperature, velocity and humidity are also factors associated with comfort, in our system we only focus on monitoring gasses.

Currently the pervasive methods of air quality monitoring make use of expensive, stationary sensors mounted at fixed locations[14], with only around 300 monitoring sites deployed across the UK [13], costing approximately £50k - £80k for the purchase and installation of a multi-pollutant site [11]. This leads to data being obtained in far-apart regions, using interpolation to determine the the air quality where no sensors are present. Furthermore, with the size and costs of the dedicated equipment such tools are not suitable for commercial use. As such they provide little guidance for people on how they can improve their living conditions in terms of indoors air quality. In this report we propose a solution to the outlined problems with stationary gas sensors, making use of the air quality index and criteria described.

Through the use of cheap, Bluetooth enabled IoT devices equipped with gas sensors monitoring the before mentioned gasses, we present a scalable system allowing for location-dense data collection, as well as personal use for standard consumers. By creating a device that can be used as a stationary station, or used on-the-move, users who would opt-in would also contribute to data collection in locations where sensors are not deployed as they travel with the device on their person. While such sensors might not be as accurate as the expensive, highly specialized devices, by scaling the system, sensor fusion methods can be applied to increase the accuracy of measurements. The readings are likely to not be as accurate even with the employment of various optimization techniques, however accuracy might not be as dependent when attempting to identify pollution "hot-spots" as we're mostly looking for comparisons in readings over time, and in different locations.

Deploying a system such as this allows for vastly location-wise denser data to be collected. This data can be used by make more informed decisions targeting "hot-spots" more affected by pollution through policies such as for example traffic diversions during specific hours. Furthermore, lowering the cost of the device allows for consumers to obtain a way of determining the most harmful sources of air pollution within their households and make alterations to the environment, helping them to improve their health and well-being.

In this report we discuss our design choices such as sensor selection and the data pipeline design making use of the Google Cloud and their SQL based distributed database system BigQuery, as well as how the system was implemented. We discuss challenges faced in the project, evaluate our measurements against data obtained from stationary sensors and evaluate the system's performance in various aspects such as power- and memory-efficiency.

1.1 Related Work

With air pollution, as well as IoT being a very popular field of research, a large amount of work has been done in the field. Studies such as [5] [2][14] looked at using IoT or monitoring networks to determine indoor air quality which is a large concern for the well-being.

Devarkonda et. al. [6] proposed a very similar solution as we did in our project. They suggested the use of mobile sensors in metropolitan areas to obtain a more fine-grained map of measurements through the distribution of larger quantities of devices to users and public transportation vehicles. Their data pipeline and visualization made use of the Google Cloud to store their data, but chose different storage components and visualization tools to process it. Their implementation focuses solely on the collection of pollutants affecting outdoor areas, while we collect additional data needed to gauge the indoor air quality as well.

2 DESIGN

2.1 Sensor Choice

For the gas or particle sensors used in our system, we chose to use all three sensors provided - the Grove Multichannel Gas Sensor, the Adafruit SGP30 Gas Sensor and the Waveshare Dust Sensor.

The SGP30 provides us with data that can be used for indoor air quality monitoring (TVOC, eCO2), the Multichannel Gas Sensor provides readings used in AQI calculations for NO2 and CO, as well as other gasses that can be associated with air pollution. Lastly, the dust sensor provides measurements of PM2.5 (fine particulate matter) which is part of the AQI calculations.

While the SGP30 sensor might not provide relevant data for outdoors air pollution, we decided that having additional options for indoor air monitoring outweighs the additional cost of the system in terms of power consumption and assembly cost.

2.2 Data Storage

To increase scalability at a low cost and to make use of the extensive collections of tools provided, we decided to make use of the Google Cloud services for data storage. In particular, as our data contained sensor readings and calculations associated with timestamps and locations, a database structure was determined to be appropriate. Thus Google BigQuery was chosen for the storage of our table data.

While we made use of only one development board with sensors in our project which uploaded its readings, BigQuery supports far greater amounts of data allowing expansions to large amounts of devices uploading to the same databases. Furthermore, through the use of Google Storage we provide a base layer of security, as the Google platform provides file-level encryption through their key-management service[7], as well as secure upload channels to their servers.

We chose to store data in two different tables - one for raw sensor readings and one for AQI calculations. We chose AQI as our method of measuring air quality. It is the most widely used standard and features usage of data provided by our sensors.

2.3 Android Application

The Android device serves as a bridge, as well as a data processing unit that sits between the sensors and the cloud. It receives data from the sensors, processes it and uploads it to the cloud. Location data read from the phone has small accuracy errors which can present difficulties when attempting to group data during analysis. Therefore, in our design of the Android application the user is able to specify a "location tag" with which data is uploaded is tagged for ease of grouping the sensor readings. The rate of upload for the sensor data is every 3 minutes in stationary mode and 30 seconds in moving mode, while the rate of upload for AQI data is every 9 minutes in stationary and every 1.5 minutes in moving mode. The mode of operation is chosen by the user through the user interface.

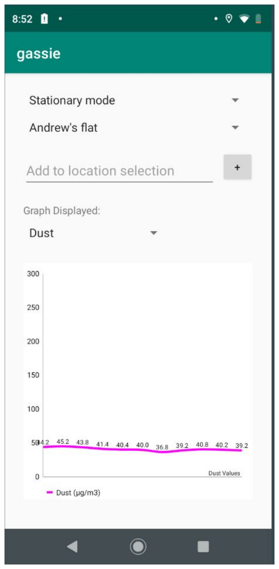


Figure 1: Screenshot of the Android application.

Furthermore, the application provides the users with real-time graphing of the sensor data where users can choose which reading they want to monitor. Figure 1 shows a screenshot of the Android application.

2.4 Visualization

2.4.1 Data Studio. Allowing for seamless integration of data from BigQuery, as well as extensive support for custom functions, we chose Google Data Studio to visualize our data. It was chosen over creating custom graphs on a separate webpage hosted on a VM due to the close integration with BigQuery where data can be imported through the Google Console interface. Figure 2 shows examples of graphs using AQI measurements, as well as the location selection feature.

2.4.2 Google Maps. Google Maps was chosen as our map platform of choice, again due to the simplicity of querying data from BigQuery and displaying it on the map. We chose to create a map where the latest recorded data of the sensor is shown alongside the AQI measurements and the date on which the data was recorded. The users can query a specific area by drawing a rectangle on the map - this displays the AQI readings for that area. Figure 3 shows an example of the map usage.

2.5 Notification Service

To alert users we have created a email-notification system allowing users to subscribe to the email list to be notified when the AQI severity level of our developed air quality monitoring system exceeds a chosen level. Figure 4 shows the subscription website for the service.

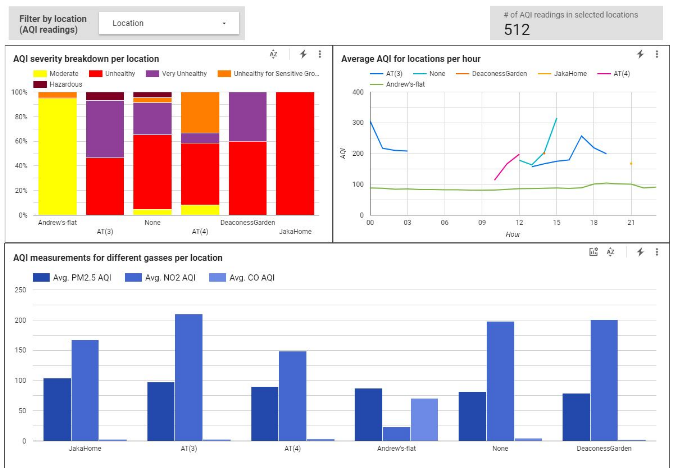


Figure 2: AQI data analysis in Google Data studio allowing for location filtering.

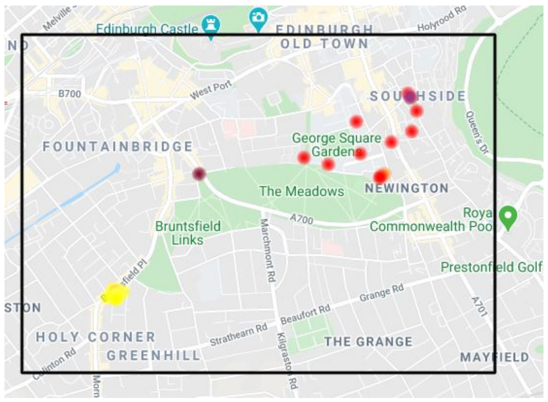


Figure 3: Google Maps with colour-coded indicators showing the AQI levels in the rectangle-marked area.

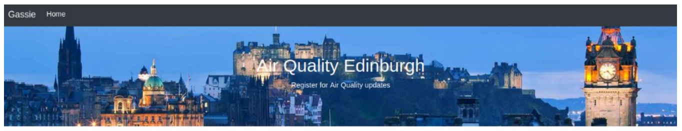
Currently, due to working with only a single air quality monitoring device, the service will notify subscribers when the readings of said device are high enough. As each reading is tagged with location data, were we to deploy several devices in different locations, location based subscription would be possible.

3 IMPLEMENTATION

3.1 Data Pipeline

The data collected from the sensors is to be processed and store in two BigQuery tables and used for visualization and notification purposes. It follows the following pipeline:

(1) Data is collected by sensors and sent to the Android device via Serial Bluetooth every 10 seconds
(2) The android application receives the data, stores it in memory. It averages the readings and calculates the AQI in set intervals which depend on whether the application is in "stationary" or "moving" mode. The data is tagged with date-time and location data and uploaded to Google Storage through Firebase.
(3) Firebase authenticates the users wanting to upload through their Google Account and secures the upload to the Google Storage bucket.
(4) When in the initial bucket, a "streaming" function determines if the file is valid and transfers the data into either the "success" or "error" bucket, as well as inputs the data into BigQuery upon success. Sensor and AQI data are inserted into two separate BigQuery tables.
(5) The data in BigQuery is queried by the end-point applications: Google Maps, Data Studio and the Notification Service.



To register for email alerts about air quality from our sensors please enter your email below:

Email address:

Enter email

We'll never share your email with anyone else.

Select level of AQI to be notified at:

Unhealthy for Sensitive Groups

Submit

Figure 4: Notification system subscription website.

3.2 Sensors and Development Board

As said, we made use of all the sensors available and connected them to the FRDM-K64F board using i2c connections. We used an additional Grove base shield to gain access to additional i2c connection slots.

All sensors aside from the Dust (PM2.5) sensor, as well as the Bluetooth module, had tried and tested mbed libraries available for configuration and use. Thus, most of the sensor setup was done using existing libraries, adapting them to our needs. The Dust sensor firmware was adapted from their example usages for the Arduino board to match the mbed functions available for our board.

We poll the sensors for data every 10 seconds, sending the data to the Android device through the HC-06 Bluetooth module using a UART serial connection. The 10 second interval can however be easily adjusted through minor modification to the configuration of our firmware if we wanted to limit or increase the amount of data being sent.

3.3 Android Application

The main purpose of the android application is to act as a relay between the sensors and the cloud, as well as to process the data into a usable format. The serial Bluetooth connection with the HC-06 Bluetooth module is established through the use of the android-serial-bluetooth library[9]. It establishes the connection with the module and receives messages on a separate thread, but requires previous pairing of the phone and module.

As the data is received every 10 seconds, if the application is in "stationary" mode, the sensor data is averaged and uploaded every 18 readings received (3 minutes), while the AQI data is calculated and uploaded every 54 readings received (9 minutes). In "moving" mode the times are shortened to 30 seconds and 1.5 minutes respectively. The user is free to switch between the two modes through a dropdown menu.

The AQI calculations make use of the 9 or 1.5 minute averages of sensor data. They are done using the equations and gas breakpoint values featured in the AQI Techincal Assistance Document released by the U.S. Environmental Protection Agency [4]. Equation 1 shows the formula used, where I_p is the AQI value, I_{Hi} and I_{Lo} the AQI value breakpoints in the bracket corresponding to BP_{Hi} and BP_{Lo} - upper and lower boundaries for concentration breakpoints. C_p is the truncated concentration of the pollutant for which the AQI is being calculated.

$$I_p = \frac{I_{Hi} - I_{Lo}}{BP_{Hi} - BP_{lo}}(C_p - BP_{Lo}) + I_{Lo} \tag{1}$$

When crafting the sensor data message to be uploaded, a timestamp of the current time obtained from the phone, as well as the user-selected location tag are added as as separate columns to be inserted into the database. In addition to those, when uploading AQI data the message also contains the current latitude and longitude of the Android device.

To upload data to Google Cloud Storage, the application is linked to a Firebase project and makes use of the Firebase Android API to upload the data. Upon launching the application, the user is asked to authenticate with their Google account which is later used for authentication with Firebase.

Lastly, the application uses the MPAndroidChart library[10] to perform real time graphing of the sensor data. Upon receiving data from via Bluetooth, a data point of the pollutant selected through the appropriate dropdown menu (Figure 1) is added to the graph.

3.4 Firebase and Google Storage

As mentioned, the Firebase API is used as a means to upload data from Android to Google Storage. The Firebase project is linked with a Google Cloud Storage Bucket into which data is uploaded. Firebase provides extensive support for Android, enabling simple and secure upload to the cloud.

The Firebase project furthermore allows for custom security rules to be enforced on who can upload data. Thus we are able to enforce authentication of users through their Google Accounts to prevent unauthorized data upload, potentially flooding the system with counterfeit data.

The use of Google Storage for our data also alleviates safety concerns, as Google automatically encrypts all of the data stored using their Key Encryption System.

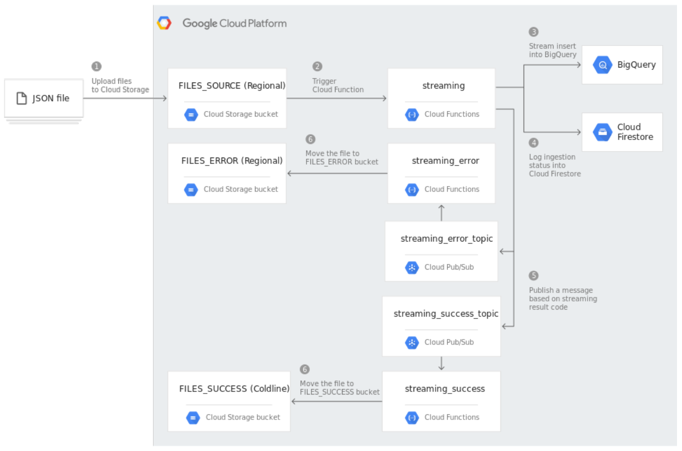


Figure 5: BigQuery streaming architecture. Source: Google[8]

3.5 BigQuery

After the data is uploaded the google cloud storage bucket, it is parsed and inserted into the BigQuery tables through the use of Google Cloud functions[8]. The "streaming" function evaluates whether the uploaded data matches the BigQuery table schema, triggering the "streaming_success" function if so, and the "streaming_error" function otherwise. The functions are triggered using a Publisher/Subscriber model available on the Cloud. The streaming method uses Firestore and Cloud Logging to log the outcome of the streaming function. An overview of the architecture can be seen in Figure 5.

The "streaming_success" function moves the uploaded JSON data file to a bucket where successfully inserted data is collected, while the "streaming_error" function moves the data to the opposite bucket which collects data that triggered an error. This allows for easier debugging and data review later on.

For storage, we use two BigQuery tables, one containing the AQI calculations, location tag, longitude, latitude and a timestamp. The second table contains the raw sensor readings alongside a location tag and timestamp.

3.6 Google Maps

The longitude and latitude data stored in the AQI table was used to visualize the readings on a map. Using the JavaScript API to query BigQuery for data, AQI readings for specific areas can be obtained. By allowing users to draw rectangles on the map, the coordinates of the corners of said rectangles are used to determine the area in which AQI readings should be shown on the map.

A query is then constructed using those longitude and latitude values as boundaries, returning the latest data for those locations, rounded to 4 decimal points to account for location data inaccuracy. The data is then plotted onto the map using colours suggested by [4], annotating the markers with AQI and date data (Figure 3 shows an example of the map). For visualization purposes the markers are resized to match the “zoom” level of the map in order to reduce clutter when viewing the map from different altitude levels.

While the map is fully functional, in the scope of this project we were not able to get the map application verified by Google. Thus only users with access to the Cloud project can query data from BigQuery using the map. As such when viewing our DataStudio report, the map features cannot be made available to users until the application is verified.

3.7 Data Studio

Google Data Studio was used for the visualization of the data collected. It was linked with both BigQuery tables, importing the data. Making use of the datetime format timestamp, values were plotted based on their time of recording, as well as transformed to perform averages over different periods of time. Furthermore, the date values allowed us to make use of the “time filter” option to allow users to view the data over a selected period of time.

We used the location tag inputted by users on the Android application to group the data. Thus users are allowed to filter the data by location, as well as compare different locations.

The Google Maps visualization was also embedded into our Data Studio report.

The Data Studio report is, as of the time of writing, available to view at: https://datastudio.google.com/reporting/7093ff73-fe37-4056-9988-c3f4ff4721d2.

3.8 Notification System

The notification system is hosted on a Google Virtual Machine for high availability, as well as to make use of the scalability of the Google Platform. It hosts a web-app (see Figure 4) where users sign up for notifications.

The emails themselves are sent using a Python script which runs on the VM, using a Python client library to access data in BigQuery. It sends out emails to users subscribed to the service if the recorded data exceeds the threshold they subscribed for. The service is ran on the VM every hour using CRON for scheduling, notifying users.

To prevent SQL injection attacks against the web-app prepared statements are used for querying. Furthermore, special HTML characters are escaped when displaying user provided information, eg. their emails.

4 RESULTS AND EVALUATION

4.1 Cost

The cost of a single sensor equipped unit is approximately £105. Counting in the assembly and cloud functionality costs, this still produces a reasonably affordable price for an average consumer. The cost calculations assume that the consumer would be able to use their own mobile device when using the system.

In the case of wide-spread stationary deployment, each device would need to be equipped with a dedicated Android phone running the needed software. Were the system to be deployed in such a manner, a different relaying technique might be needed to reduce the assembly costs.

4.2 Efficiency

4.2.1 Power usage. We measured the current draw of our sensor system through the use of a voltmeter measuring the draw while the device was active. The current draw ranged from 0.22A to 0.27A amounting to a maximum battery life of 18.5 hours on a 5000mAh batter. Testing the device in the real world with a 5000mAh power-bank we were able to power the device for 14 hours, showing sufficient battery life for personal usage.

4.2.2 Memory usage. Compiling the sensor and Bluetooth firmware, we found that it uses a total of 84.7KB of program flash memory (8.27% of the board’s flash memory) and 13.128KB of static RAM (5.11% of the device’s). This gives us plenty of room for firmware updates and new features would we want to add these to our existing hardware, or upgrade the device with other sensors or tools.

4.2.3 Bandwidth usage. Within the stationary mode, the application uploads sensor data every 3 minutes and AQI data every 9 minutes. The rate of upload is 6x faster in moving mode (0.5 minutes and 1.5 minutes). Each sensor data message contains on average 185 Bytes of data, while the AQI data size is 146 Bytes on average. This amounts to approximately 4.56KB of data being sent per hour while stationary, and approximately 27.38KB while moving. With current mobile network bandwidth capabilities, this should present no issue and incur very low monetary cost overhead.

Furthermore, the small data size and adjusting the frequency of uploads reduces the costs of maintaining the data on the cloud. While that would not be the case when the devices would be deployed in larger quantities, during the course of this project we uploaded approximately 500 AQI readings and 1500 sensor readings and performed an extensive amount of queries and visualization, but have not incurred any costs as of the time of writing.

4.3 Data Evaluation

4.3.1 Initial Data and Optimization. Our initial data collection showed that the readings of our sensors, in particular the NO2 values were significantly higher and more volatile than expected. Using a separate Arduino board to make use of the SGP30’s calibration libraries, we managed to improve and stabilize the readings. Figure 2 shows an example the fluctuations in the readings in the AQI severity breakdown graph. All of the readings in locations aside from “Andrew’s-flat” show extremely high and volatile AQI values, which were obtained pre-calibration. As can be seen from the graph however, looking at the “Andrew’s-flat” data the readings were lowered and stabilized.

4.3.2 Data Accuracy. While the data did stabilize post-calibration, testing against a stationary monitor present in Edinburgh at St. Leonadard’s land showed that they were still somewhat inaccurate, but more in line with the actual air quality as before. Figures 6, 7, 8, 9 show the plots of our data against the data collected at the monitoring station.

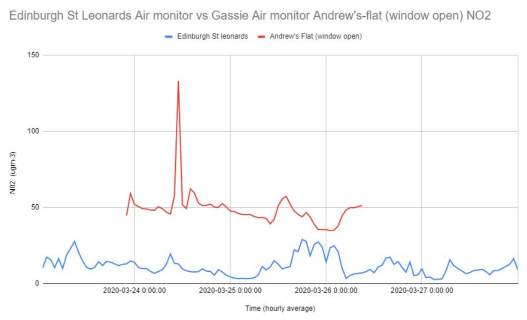


Figure 6: NO2 comparison between "Andrew's-flat" and St. Leonard's land.

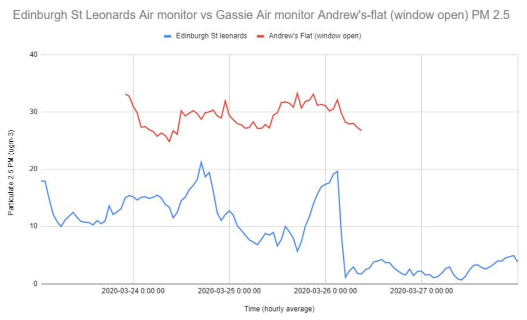


Figure 7: PM2.5 comparison between "Andrew's-flat" and St. Leonard's land.

We can observe that while our data is higher than the actual readings, we still obtain readings in a reasonable scale and can use them to observe relative changes in pollutant presence within the air. Furthermore, as the PM2.5 values were measured inside a building, the dust levels were likely higher, inflating the values.

Lastly, we observe that after the calibration of the sensors the CO readings deteriorated, but they are still not far out of the boundaries, making the data again useful for tracking changes in levels. Additionally, the values do not differ enough to cause a significant change in the AQI calculations, or in our case, do not affect them at all.

4.3.3 Data Obtained. The stored data was used for visualization in Google Data Studio. Figures 10, 11. 12, as well as 9 show different visualizations done in Data Studio.

All of these can be used by consumers or researchers to evaluate the air quality in different locations. Over time readings and averages shown in Figure 10 can be used for determining which pollutant affects air quality of a region the most, as well as provide feedback on improvements or deterioration. Figure 12 shows how individual gasses are visualized - were the device to be used for indoors air quality monitoring, consumers would be able to determine the readings of TVOC and CO2.

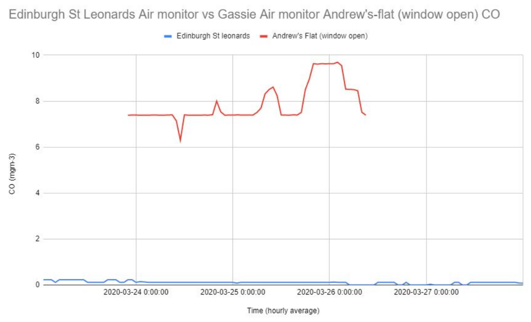


Figure 8: CO comparison between "Andrew's-flat" and St. Leonard's land.

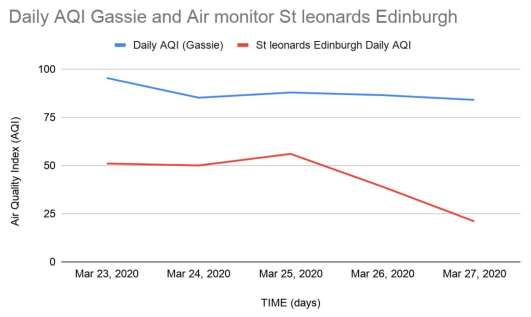


Figure 9: AQI comparison between "Andrew's-flat" and St. Leonard's land.

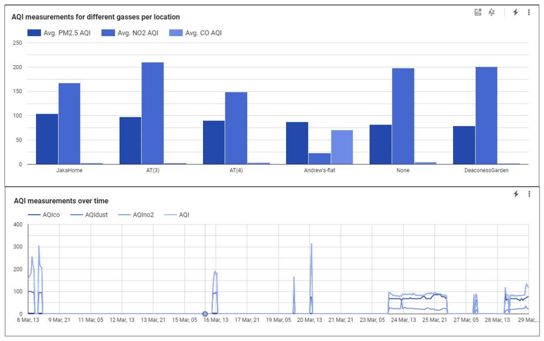


Figure 10: Average and over-time AQI visualization. Over time graph can be filtered by location.

4.4 Challenges

Sensor Accuracy. As was mentioned before, while we managed to improve the sensor accuracy through calibration, the accuracy was still not satisfactory. Employing averaging helped smooth out the data and remove a large amount of noise, however overall the readings were less accurate than we hoped for. With the addition of several sensors located near each-other, the data accuracy could be improved. We noticed that calibration plays a large role in the readings - in our case the CO readings deteriorated after calibration,

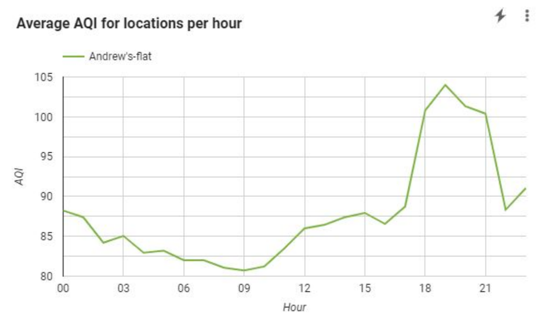


Figure 11: Hourly average AQI for "Andrew's-flat".

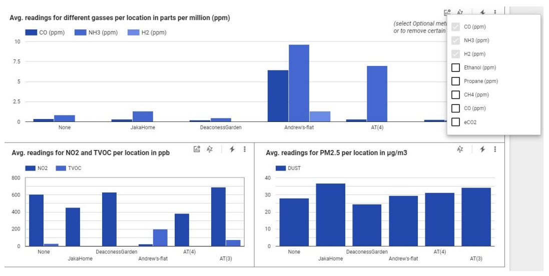


Figure 12: Visualizations of averages for different pollutants.

but NO2 improved. If we combined different sensors which were perhaps calibrated under different conditions, the accuracy would likely improve.

Device Location. While we alleviated some errors in the location noise through custom user-tagging through text input, as well as rounding the longitude and latitude readings, that still requires either a compromise in accuracy or user-input which is prone to error. Thus using location automatic location tagging based on device location would be a preferred approach for future iterations. This data could be polled from tools such as Google Maps.

Encryption of Sensor Data. While we attempted to provide symmetric encryption from the sensors to the mobile phone, we stumbled upon challenges when attempting to implement matching encryption/decryption protocols on both ends. We deemed security when uploading sensitive data to the cloud (location) to be more important than the encryption of the Bluetooth connection where only sensor readings are sent to the android device locally. Thus we postponed it due to the implementation difficulties experienced.

5 POSSIBLE EXTENSIONS

While the current state of the system is satisfactory for our goals of creating a proof-of-concept prototype, there are several extensions and future work that the system allows for.

Location Based Notification. As we were working with only one sensor-system during the course of development, it was decided to build the notification system such that users will be alerted when any reading of that device exceeds threshold levels. However, when expanding the system to multiple devices that would no longer be an optimal solution. Thus we would use location-boundary based querying where users would subscribe to notifications that come from sensors in a select area. The system would use a similar style selection method as used in our Google Maps visualization system where users would draw a rectangle indicating the area they're subscribing to.

Privacy with Multiple Devices. Extending the system to make use of multiple consumer-level devices comes with the challenge of privacy concerns. Thus all data would have to be anonymised and made available to only the user unless they chose to share the data for either research purposes or to make it available to the general public. Thus each device would be associated with an unique ID, as well as a flag stating what level of privacy the user chooses.

This would be implemented through adding two fields to our BigQuery databases - ID and PrivacyLevel. Furthermore, individual Data Studio reports would be created where users would be able to view and analyze their private data.

Indoor Air Quality Calculations. Currently our system only provides calculations for the AQI, which is traditionally used for outdoors air quality measurement, requiring users to interpret raw TVOC and CO2 readings. In future iterations we would create a third BigQuery table featuring an indoor air quality measurement schema to ease the interpretation of data.

ACKNOWLEDGMENTS

I would like to thank my teammate Andrew Lindsay for all his work on this coursework while juggling his dissertation, as well as just being a great partner in general.

REFERENCES

[1] Sabah Ahmed Abdul-Wahab, Stephen [Chin Fah En], Ali Elkamel, Lena Ahmadi, and Kaan Yetilmezsoy. 2015. A review of standards and guidelines set by international bodies for the parameters of indoor air quality. *Atmospheric Pollution Research* 6, 5 (2015), 751 – 767. https://doi.org/10.5094/APR.2015.084
[2] Sherin Abraham and Xinrong Li. 2014. A Cost-effective Wireless Sensor Network System for Indoor Air Quality Monitoring Applications. *Procedia Computer Science* 34 (2014), 165 – 171. https://doi.org/10.1016/j.procs.2014.07.090 The 9th International Conference on Future Networks and Communications (FNC'14)/The 11th International Conference on Mobile Systems and Pervasive Computing (MobiSPC'14)/Affiliated Workshops.
[3] U.S. Environmental Protection Agency. February 2014. Air Quality Index, A Gude to Air Quality and Your Health. https://www3.epa.gov/airnow/aqi_brochure_02_14.pdf
[4] U.S. Environmental Protection Agency. September 2018. Technical Assistance Document for the Reporting of Daily Air Quality – the Air Quality Index (AQI). https://www3.epa.gov/airnow/aqi-technical-assistance-document-sept2018.pdf
[5] S. Bhattacharya, S. Sridevi, and R. Pitchiah. 2012. Indoor air quality monitoring using wireless sensor network. In *2012 Sixth International Conference on Sensing Technology (ICST)*. 422–427.
[6] Srinivas Devarakonda, Parveen Sevusu, Hongzhang Liu, Ruilin Liu, Liviu Iftode, and Badri Nath. 2013. Real-Time Air Quality Monitoring through Mobile Sensing in Metropolitan Areas. In *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing* (Chicago, Illinois) *(UrbComp '13)*. Association for Computing Machinery, New York, NY, USA, Article 15, 8 pages. https://doi.org/10.1145/2505821.2505834
[7] Google. 2020. Data encryption options. https://cloud.google.com/storage/docs/encryption

[8] Google. 2020. Streaming data from Cloud Storage into BigQuery using Cloud Functions. https://cloud.google.com/solutions/streaming-data-from-cloud-storage-into-bigquery-using-cloud-functions
[9] harry1453. 2020. android-bluetooth-serial library. https://github.com/harry1453/android-bluetooth-serial
[10] Phillip Jahoda. 2020. MPAndroidChart library. https://github.com/PhilJay/MPAndroidChart
[11] Diane Mooney. 2006. A Guide for Local Authorities Purchasing Air Quality Monitoring Equipment. https://uk-air.defra.gov.uk/library/reports?report_id=386
[12] World Health Organisation. 2020. Health Topics in Air Pollution. https://www.who.int/health-topics/air-pollution
[13] UK AIR Air Information Resource. [n.d.]. Monitoring Services. https://uk-air.defra.gov.uk/networks/
[14] A. Tapashetti, D. Vegiraju, and T. Ogunfunmi. 2016. IoT-enabled air quality monitoring device: A low cost smart health solution. In *2016 IEEE Global Humanitarian Technology Conference (GHTC)*. 682–685.