Geo-Spatial Air Quality Monitoring System

Yashvi Chawla
School of Informatics, University of Edinburgh
yashvi.chawla@gmail.com

ABSTRACT

With increasing air pollution, the quality of air has been excessively degrading since the last few years. Air quality has become a major concern today due to it's adverse impacts on human health and to the ecosystem as a whole. Thus, there is a need for efficient air quality monitoring systems to capture the trends in air quality and help citizens understand the different levels of pollution they are experiencing in their everyday lives. In this paper, we propose and develop a full stack IoT based Geo-Spatial air quality monitoring system. Different sensors are used to collect the pollutants in the air and assess the air quality. A cloud based analytics pipeline is created for analysing and visualizing the collected pollutant concentrations. An air quality index (AQI) is computed from these pollutant concentrations and is used to notify the users about the quality of air in their location.The system is capable to handle trends in air quality in real time.

ACM Reference Format:
Yashvi Chawla . 2020. Geo-Spatial Air Quality Monitoring System. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Air pollution is a major global challenge and is estimated to cause a death of 7 million people every year. As reported by World Health Organization(WHO), air pollution is the world's fourth leading fatal health risk as more than 80% of the population today lives in areas of degrading air quality . Long term exposure to such degraded air quality results in serious health issues and increases the risk of fatal diseases such as lung cancer, heart diseases, stroke and chronic bronchitis. Despite a global problem, there is not much opportunity for people to understand the pollution level they experience in their everyday lives.There is a need for efficient air quality monitoring systems which are capable of measuring and analysing the harmful pollutants in the air. Currently, large and expensive monitoring stations which have been permanently deployed and professionally maintained at a small number of locations are used to perform air quality monitoring across the globe. However, the installation of such huge infrastructures demands a significant amount of time and money. Moreover, the monitoring data is not available in real time and is very sparse. Recently, more research is being done to investigate IoT based low cost sensor strategies to perform more efficient and widely deployed air quality monitoring.

Advancement in technology have led to the development of smaller, portable and low cost sensors which can measure and report air quality in real time. Most researchers are targeting the newly emerged capabilities of IoT, Big Data and cloud computing to address the air quality monitoring globally. These projects mainly focus on stationary or mobile wireless sensor networks to collect and measure the harmful pollutants in air. The GSMA in collaboration with the Royal Borough of Greenwich has developed an air quality monitoring initiative in London based on IoT and Big Data technologies to help citizens understand the levels of pollution they are exposed to in their daily lives [1]. Similarly, the OpenSense project conducted in Switzerland, had all the sensors deployed on a moving tram for a wider coverage of the city [2].

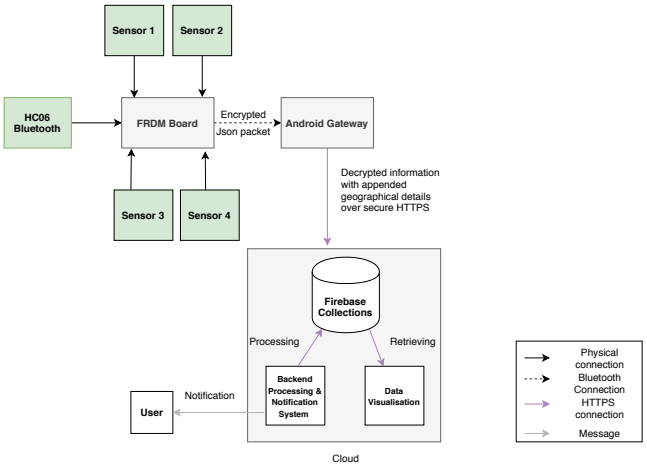

Figure 1: Architectural Overview of the Prototype

In this paper, we propose and develop a real time full stack IoT based Geo spatial air quality monitoring system. The project aims at collecting data for longer periods of time at different locations in the city of Edinburgh and perform analysis on the spatio-temporal variation of the pollutants measured using cloud solutions. Moreover, the system aims at notifying the users subscribed to our monitoring service the changes in air quality when it exceeds certain limits. Other than analysis and visualization of collected data, we also discuss the security aspects of the end to end prototype developed. Based on this, the objectives of our project are:

- Build a firmware for our IoT system.
- Build a simple Android Application that enables sending the measurements collected by the sensors to the cloud.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- Build a cloud based analytics pipeline and a visualization dashboard for spatio temporal data processing and real time visualization of air quality.
- Build a notification system to alert the users about the air quality in their location.
- Use encryption-decryption techniques to ensure secure transfer of data from the embedded device to the cloud.

2 PROTOTYPE

This section briefly outlines the different segments of our full stack IoT system. Figure 1 gives an architectural view of our air quality monitoring system.

2.1 Firmware

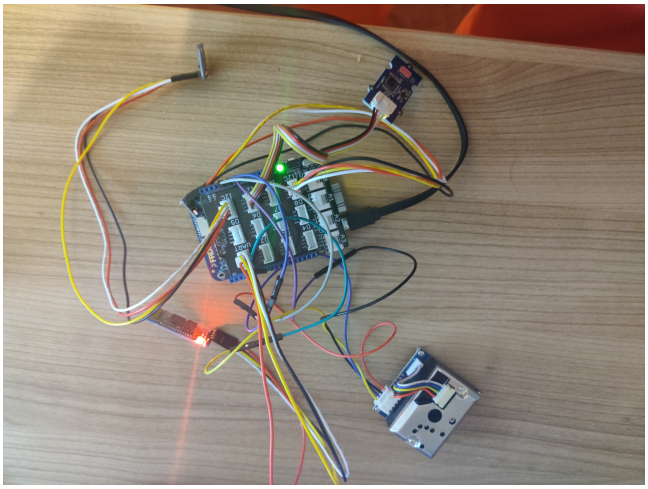

Figure 2: FRDM-K64F Development board with different sensors

In this project, a FRDM-K64F Freedom Development Platform is used to deploy our embedded application that reads from different sensors attached to it. The sensors are not capable enough to process the data by themselves as they have limited resources and thus require a processing platform to handle data collection and transmission efficiently. The FRDM-K64F platform is a set of software and hardware tools ideal for rapid prototyping of microcontroller based applications. The K64F platform is build on an ARM Cortex-M4 Core with a 1MB flash memory, 256KB RAM, a full speed USB controller, Ethernet Controller and analog and digital peripherals [3]. The onboard interface includes a six-axis digital accelerometer & magnetometer, RGB LED, add-on Bluetooth module, add-on RF module, and Ethernet. Different sensors are attached to the development board to periodically scan the environment and measure the pollutants harmful for human health. The sensors used for air quality monitoring include:

Grove - Multichannel Gas Sensor: Multichannel Gas sensor is an environment detecting sensor with a built in MiCS-6814 sensing element to detect harmful gases [4]. It uses I2C interfacing to connect to the development platform and is capable of giving only approximated gas concentrations. We use multichannel gas sensor to detect Ammonia(NH3), Nitrogen Dioxide(NO2) and Carbon Monoxide(CO).

Adafruit SGP30 TVOC/eCO2 Gas Sensor: The Adafruit gas sensor is mainly used for indoor air quality monitoring as it can detect a number of Volatile Organic Compounds (VOCs) and H2 [5]. This sensor also uses I2C interfacing to connect to our micro-controller and is limited in its calibration capabilities. The sensor is used to measure eCO2 (equivalent calculated carbon-dioxide) and TVOC (Total Volatile Organic Compound) concentration.

Optical Dust Sensor: Dust sensor [6] is used to measure dust and smoke concentration which are fine particles larger than 0.8μm in diameter. It gives the concentration of dust in ug/m^3.

BME280 **sensor**: Adafruit BME280 sensor is an environmental sensor used to measure Temperature, Barometer Pressure and Humidity in the atmosphere [7] . The sensor works with both I2C and SPI interfacing to exchange data with micro-controller. It is a low cost sensor and gives an accuracy of +1% to +3% when measuring temperature, pressure and humidity.

All the sensors are connected to the development platform using I2C interfacing and are implemented using various C++ and Mbed libraries. Figure 2 depicts the FRDM-K64F development board with the attached sensors. The sensor data is collected every 5 seconds and is transmitted from the device to the Android application over Bluetooth Communication. An HC-06 Bluetooth Module operating on Bluetooth 2.0 communication protocol is used for wireless data transmission from the device to the android application [8]. The sensor data is converted to JSON format before transmission. This choice facilitates easy storage and access of data from the database in the form of key value pair. It further makes data processing much easier. The data is collected after a delay of 5 seconds which saves power consumption and processing resources without missing any trends during sudden changes in data. A portable battery is used to power the device and minimal data processing is performed on the device to save on device resources. The data is encrypted using an RSA public key to ensure secure transfer of data from the embedded device to the Android application.

2.2 Android Application

The Android application acts as a gateway between the embedded device and the cloud and enables forwarding the measurements collected by the embedded system to the cloud via Bluetooth communication. Android-bluetooth serial library is used to both read and send messages to the embedded device. To enable communication over bluetooth certain permissions need to be enabled to inform the OS that the application has access to bluetooth functionality. These permissions include:

- android.permission.**BLUETOOTH**
- android.permission.**BLUETOOTH_ADMIN**
- android.permission.**ACCESS_COARSE_LOCATION**

The android application is designed with a simple user interface in android studio using Recycler View Class and View Bindings to

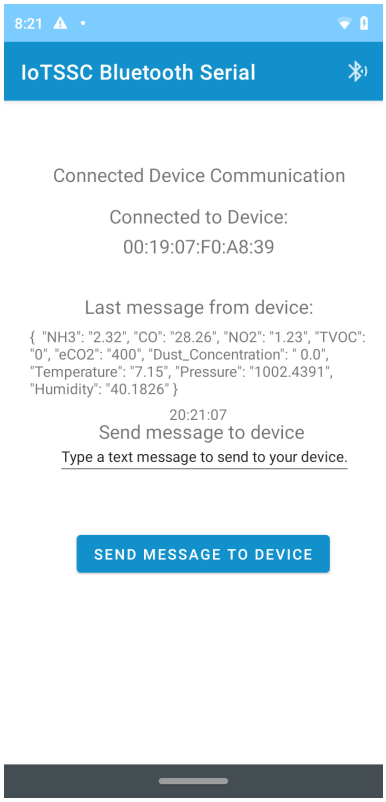

Figure 3: Android Gateway

detect and display Bluetooth devices. Once a device is connected to the phone, an *onConnected* callback of bluetooth serial library is invoked which records the mac address of the connected device and set listeners for handling messages to/from the device. When a message is received from the device, it is first decrypted using a RSA private key. The android Location Manager class is used to append geo-location information to the message either via GPS or via Network Provider and the android Calendar Class is used to append timestamp information to the message [9]. Android Firebase API is then used to connect to Google Firestore Database to store collected sensor data in the key value format [10]. The connection from Android to Firebase is authenticated and made secure via configured keys. Minimal processing is done on the app end to save resources.Figure 3 depicts the overview of our Android Application.

2.3 Cloud

We implemented a Google cloud based analytics pipeline for data analysis and visualization of collected sensor data. We chose Firebase to store the collected data as it is flexible, secure, easy to integrate with android and already linked with google cloud. The data is stored in Firestore database in NOSQL format in the form of collections and documents. Figure 4 illustrates the overall view of the database format used in FireStore. We make a collection named *sensors_data* wherein each reading from the device is stored as a unique document name in the form of a date and timestamp, for instance 2020 : 03 : 30_11 : 09 : 54. This data format facilitates in the

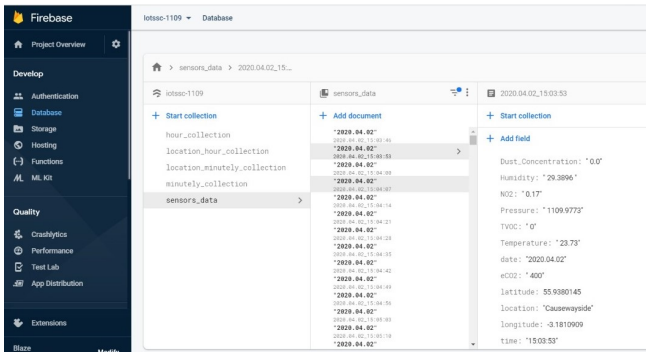

Figure 4: Firebase Database Format

time series analysis of the sensor readings. The different pollutant concentrations and the geo-location information are stored as key value pairs inside the document. This further makes the accessing and processing of data easier for analysis. A Virtual Machine (VM) Instance is created on the Google Cloud Compute Engine to run a python file in background for elaborate data processing and analysis. We make a secure connection to Firebase using Firebase Admin library and the connection is authenticated using JSON certificate credentials generated during the setup. Once a connection is set, all readings from the sensors_data are fetched and converted to minutely and hourly readings for better air quality analysis. An Air Quality Index (AQI) is used as a measure to report air quality. It is a standard technique used by government organizations to communicate to people how polluted the air currently is and what are the health risks associated with it.

AQI Algorithm

Air quality index is computed using air pollutant concentrations collected over a specified period of time usually hourly, 8 hourly or 24 hourly. For our project, AQI index is computed on an hourly basis using 4 pollutants: Ammonia (NH3), Nitrogen Dioxide (NO2), Carbon Monoxide (CO) and PM2.5. We use six AQI categories, namely Good, Satisfactory, Moderately polluted, Poor, Very Poor, and Severe over a range of 0-500 to denote the air quality and its associated health impacts. The formula used to compute AQI is:

$$I = \frac{I_{high} - I_{low}}{C_{high} - C_{low}}(C - C_{low}) + I_{low} \quad (1)$$

where:
I = Air Quality Index, C = the pollutant concentration,
C_{low} = the concentration breakpoint < C, C_{high} = the concentration breakpoint > C, I_{low} = the index breakpoint corresponding to C_{low} and I_{high} = the index breakpoint corresponding to C_{high}. An individual score (Individual Air Quality Index, AQI) is calculated for each pollutant based on its concentration and the final AQI is computed by averaging the individual AQIs of all pollutants. The AQI values and the corresponding concentration breakpoints used in the formula to compute final AQI as well as associated health impacts for the identified pollutants are depicted in table 1.

Table 1: AQI and pollutant concentration breakpoints with associated health risks

AQI Category	PM2.5	NO2	CO	NH3	Health Impact
Good (0-50)	0-30	0-40	0-1.0	0-200	Minimal impact
Satisfactory (51–100)	31- 60	41- 80	1.1-2.0	201-400	Minor breathing discomfort
Moderately Poor (101–200)	61- 90	81- 180	2.1-10	401-800	Breathing discomfort to people with lung diseases
Poor (201–300)	91-120	181-280	10-17	801-1200	Breathing discomfort to people on prolonged exposure
Very poor (301–400)	121-250	281-400	17-34	1200-1800	Respiratory illness to the people on prolonged exposure
Severe (401–500)	250+	400+	34+	1800+	Serious health impacts

Notification System

We use Twilio Api to implement the Notification System which alerts all the users subscribed to our monitoring service when the AQI level excedes certain limits (beyond 100). The backend program makes a secure connection to twilio API through twilio library and configured API keys. The connection is authenticated using an authentication token and a secret key identity. We add the list of phone numbers and the message to be sent to the users in the backend program and push the details to the Twilio APi which sends the message to all the users reporting the current situation of air quality and its detrimental health impacts. This helps to signal the users which times of day and what locations to avoid if the air quality is harmful for health. This makes it easier for citizens to understand the levels of pollution they experience in their daily lives, as the monitoring data is available in real time and is specific to their location.

3 DATA ANALYSIS & VISUALIZATION

For elaborate air quality monitoring analysis, data is collected from 4 locations namely CausewaySide, Nicolson Square, Morgan Lane and Relugas Place. This data is collected continuously for several hours during different timestamps and on different dates for better space and time analysis. Almost all analysis performed such as average AQI and pollutant concentrations, peak polluted times, Hotspot locations, Previous data analysis, Correlation Analysis are implemented in the backend Python program and update in real time. The Visualization Dashboard is implemented in Html and Javascript to accommodate responsiveness and real time changes. We use Zing charts API to implement all the graphs in our visualization dashboard.

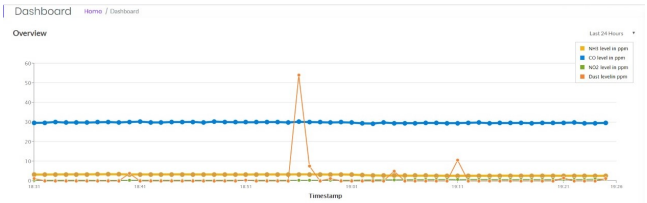

Figure 5: Average Pollutant Concentrations on minutely basis

We first analyse the average concentration of each pollutant collected for a particular location on minutely basis. This analysis can be performed for all the locations where sensors are deployed to show real time trends of the pollutants in the air as data is being collected simultaneously. All the sensor readings collected after every 5 seconds from the device are averaged and converted to minute readings and stored in another collection namely *Minutely_Collection*. The visualization dashboard connects to the firestore database, retrieves readings from the minutely_collection and shows the changes on the minutely analysis graph. Though the analysis is onerous and consumes a lot of resources, it can perfectly capture the sudden trends in air quality which might be useful for performing more elaborate forecasting analysis. Figure 5 depicts the avg concentration of different pollutants on a minutely basis.

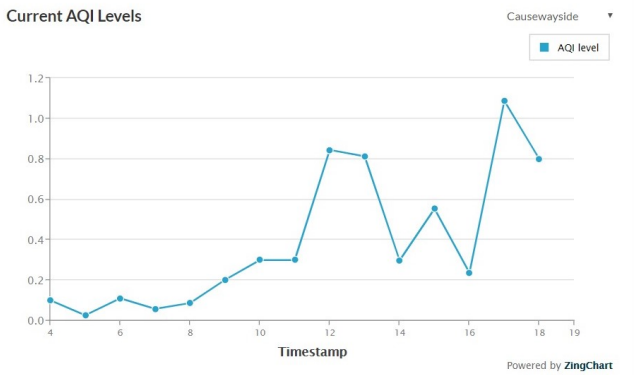

Figure 6: Average AQI index on hourly basis

We further compute average AQI index for each location on an hourly basis to help users know more about the air quality in their region. The AQI index is computed using the standard formula mentioned above from the average hourly concentrations of different pollutants. Figure 6 illustrates the avg AQI levels for each location on an hourly basis. The graph updates AQI values in real time after every hour.

We use Google's Map API to represent all the locations where data is collected. The *Location_hour_collection* in the firestore database is automatically updated with latitude and longitude coordinates whenever a new location where data is collected is encountered.

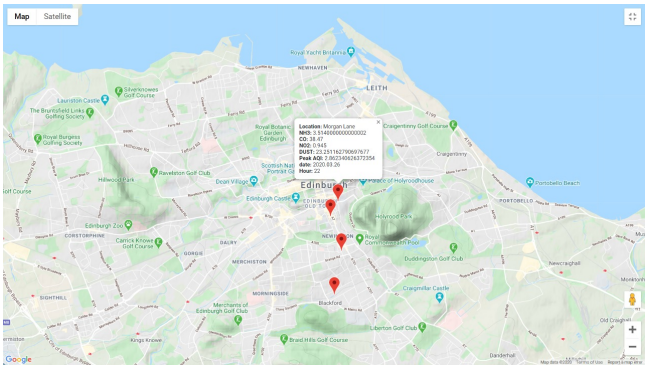

Figure 7: Different locations where data is collected with Peak AQI and peak pollution time

The collection keeps track of hourly average pollutant concentrations and AQI values for each location and detects the peak AQI value for each location over a period of time for a particular day. The peak pollution time and peak pollutant concentrations are updated based on the peak AQI values. The map processing module spans over the entire *location_hour_collection* and the different marker points on the map indicate the different locations covered with their peak pollution time, peak AQI and pollutant values on a particular day. Based on these Peak AQI and pollutant concentrations, we are able to analyse the hotspot locations giving insightful information to our users. Figure 7 shows the above analysis on Google Maps.

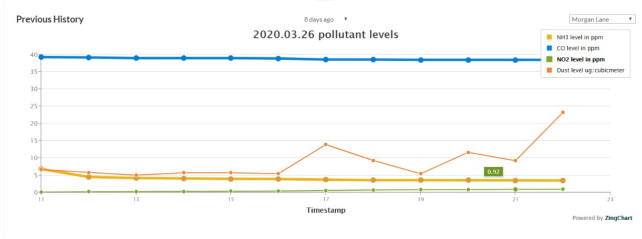

Figure 8: Historical Data Analysis

We also do historical data analysis where previous history of pollutants for all locations can be seen. This helps our users to better understand the air quality patterns in their regions. We use firebase compound queries in javascript to retrieve pollutant readings from the *location_hour_collection* based on the day and the location. We can view historical data for all locations upto as many days as we want. Figure 8 depicts the visualization graph for our historical data analysis.

Further, we use Pearson's Correlation Coefficient to perform a correlation analysis of pollutants with themselves, other pollutants and with Temperature, Pressure and Humidity to better understand the relationship between each of them. Pearson's Correlation Coefficient is a statistical formula that measures the strength between

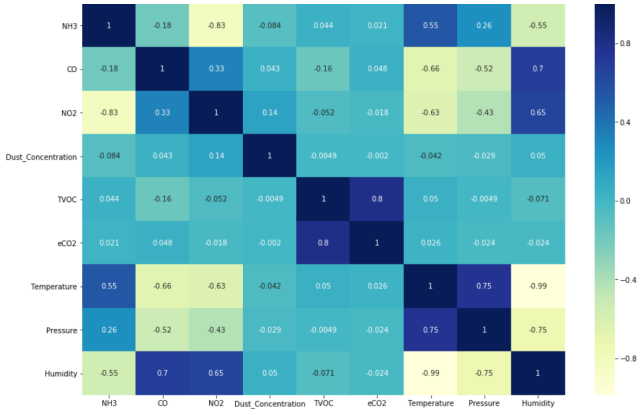

Figure 9: Correlation Analysis

variables and relationships. The coefficient value can range between -1.00 and 1.00. If the coefficient value is in the negative range, then that means the relationship between the variables is negatively correlated, or as one value increases, the other decreases. If the value is in the positive range, then that means the relationship between the variables is positively correlated, or both values increase or decrease together. The equation used to perform correlation analysis is:

$$r = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sqrt{\sum(x - \bar{x})^2(y - \bar{y})^2}} \quad (2)$$

where r = Pearson's Correlation Coefficient, x= pollutant 1 concentration, $\bar{x}$ = mean of pollutant 1 concentrations, y= pollutant 2 concentration and $\bar{y}$ = mean of pollutant 2 concentrations. It is an exploratory data analysis which helps to explain the trends in different pollutants with variations in other pollutants, temperature, pressure and humidity. Figure 9 potrays an SNS heatmap for our correlation analysis. It can be observed that Ammonia (NH3) has strong positive correlation with Temperature whereas Carbon Monoxide(CO) and Nitrogen Dioxide (NO2) are negatively correlated. Thus, NH3 measurements have more variations during high temperature than CO and NO2 measurements. We can also observe that Nitrogen Dioxide (NO2) and Ammonia (NH3) are strongly negatively correlated.This denotes that trends in Nitrogen Dioxide(NO2) measurements increase with decrease in Ammonia (NH3) concentrations.

Evaluation

To validate the accuracy of our readings, we checked the publicly available real time information about air quality in Edinburgh. We were not able to validate the readings for the exact locations which we covered as data were not available for them but we were able to do our evaluation based on the data of nearby locations. Most of the pollutant concentrations we measured were in the same range as shared by the publicly available data. Table 2 indicates the statistical readings when compared to publicly available data. We were able to analyse that our AQI values were in the same band as publicly available real time AQI values in Edinburgh but the values were much less when compared to real readings. This discrepancy

h

Table 2: Comparing measured readings against public data

Pollutant	Our Reading	Public Data Range
PM2.5	23.25	13-34
NO2	0.945	1-12
Temp	10.23	1-15
Pressure	1041	1034-1049

in results is because their AQI is computed based on highest pollutant concentration which is mostly O^3 or $SO2$. As our sensors are not capable of collecting more harmful pollutants like O^3 or $SO2$, we calculate the AQI for all our pollutants and the final AQI is computed by averaging the individual AQIs of all the pollutants. Moreover, our readings are collected mainly indoors and not in traffic prone areas which further affect the AQI reading as a whole. The highest AQI value which we measured was 24 at Nicolson Square on 27/03/2020, where the readings were computed outdoors. The readings of publicly available data were in the range 30-32 for Leonard's Street on 27/03/2020 which is the closest location to Nicolson Square whose data was publicly available.

4 SECURITY

Our embedded device has limited computing power and memory capacity which makes the hardware resources insufficient for the support of complex cryptography. However, security is an essential requirement for our system deployment as a secure communication needs to be established while sending data from our device to the android application. We use an asymmetric key cryptography algorithm, RSA (Rivest-Shamir-Adleman) to ensure data secrecy, participant authentication and non- repudiation. It is a public key algorithm wherein the public key is shared openly but the data can only be decrypted using a private key unique to the owner. We encrypt the JSON packet key and value pairs on the embedded device using RSA public key and decrypt the data packets in Android using RSA private key. This helps us to ensure that the man-in-the-middle attacks can be avoided and data can be transmitted securely from the device to the android application. A message digest cryptographic function can be implemented as future improvement to add another layer of security for our encrypted data.

Further, the transmission of data from Android to the Cloud is secure as data is transmitted over secure Https line by Firebase protocols. We make a secure connection to Firebase using Firebase Admin library and authenticate using JSON certificate generated during the setup. Our Visualization board also makes a secure connection to Firebase while accessing data via configured API keys. Similarly, while implementing our Notification System, the backend program makes a secure connection to twilio API through twilio library and configured API keys.

5 TECHNICAL CHALLENGES

This section outlines the different challenges we faced while implementing our prototype for air quality monitoring.

5.1 Device Setup

It was difficult to set up the ARM MBED command-line tool on Windows as not much resources are available for windows installation and setup. The problem was due to incompatible versions of installation packages. We were able to resolve the issue by manually changing the versions of these packages on the Requirements file of the Mbed Os folder. We further faced a difficulty in implementing our Bosch - BME280 Temperature, Pressure and Humidity Sensor. This was due to the usage of wrong MBED library for the sensor implementation. We were able to overcome this challenge by manipulating the existing BME280 library as per our requirements.

5.2 Android Workflow

Initially we tried to send data from Android Application to Cloud using PubSub methodology over HTTP bridge but while authenticating with public and private keys we got a permission denied error. We spent a lot of time figuring out what was wrong and even took our demonstrators' help but were not able to resolve the issue. The documentation provided by Google for sending data over HTTP bridge is not elaborate and definitely not suitable for beginners. Also, initially we planned to use BigQuery for data storing and analysis but there is no direct integration from Firebase to Bigquery. We thought of using a Bigquery extension provided by firebase library to send data from firebase to Bigquery. However, while sending data through an extension, the entire data was stored in a single column as a string format which made it difficult for elaborate analysis. Thus, we figured out that BigQuery was not suitable for the type of processing we wanted to achieve. We then decided to use Firestore database for storing data as it is flexible and easily integrated with android and google cloud. For elaborate real time data processing we decided to implement a backend program in Python.

5.3 Real Time Processing of Data

Though real time processing is an interesting task, but it was difficult to keep track of updating timestamps. We took inspiration from MapReduce programming approach to resolve this problem of updating timestamps as well as keeping track of previous timestamps and peak AQI values. The Visualization dashboard considers some delay before updating real time graphs. This is done to reduce frequent access to firestore database, to save resources and to avoid network congestion.

5.4 Encryption Decryption

As security is an important aspect of IoT deployment, we initially decided to implement Symmetric AES (Advanced Encryption Standard) algorithm as it is commonly used for cross platform encryption and decryption. We had a hard time implementing encryption and decryption with the AES algorithm due to different bits and byte configuration in C++ and Java as well as padding problem caused due to difference in the block sizes. We tried to solve the padding issues to match the length of the keys in both C++ and Java by using a PKCS5 padding. However, the padding approach did not work and generated a *Bad Key* error. The issue is yet to be resolved and as we were short of time we switched to asymmetric RSA algorithm to fulfil the security aspects of our prototype.

6 FUTURE IMPROVEMENTS

This section briefly outlines the improvements which can be done in our project to make it more efficient.

6.1 Data Collection Evaluation

We would like to cover more traffic prone and industrial areas in the city for data collection. This would enable us to receive better measurements which would aid in achieving better AQI results and air quality analysis. A more thorough evaluation of the results can be done, once data is collected and analysed from the major pollution sites of the city. Moreover, we would like to increase our hardware setup with multiple wired and wireless sensors so that data can be collected and processed simultaneously from different locations in the city. Currently, even though we are limited with our hardware setup and only one location data can be collected at a time, our system is still capable of handling real time analysis of multiple locations at the same time.

6.2 Added Security Features

We would like to implement Hashing in our embedded device to add another layer of security to our encrypted data. It would also help us to optimise our embedded application as hashing transforms the data into a far shorter fixed-length value or key. The added security by Hashing will help us to identify whether our data has been altered or not thus ensuring the integrity of the data transmitted.

6.3 Forecasting

With more collected data, we would definitely like to implement a time series forecasting algorithm for air quality monitoring. Time series forecasting is widely performed for real time air quality analysis wherein a model is used to predict future values based on previously observed values. We would like to implement one of the most commonly used method for time-series forecasting, ARIMA (Autoregressive Integrated Moving Average) model which are capable of capturing the trend, seasonality and noise in the data.

7 INDIVIDUAL CONTRIBUTION

The entire project was quite rigorous and demanding, but extremely interesting to work on at the same time. We were introduced to so many new technologies and it was a delight to learn every one of them. The project started on a good note wherein Tom and I, attended all the labs together and completed our firmware implementation during the lab sessions itself. However, the project was much disrupted due to unforeseen strikes and later due to covid-19 pandemic which resulted in lack of communication as a team. We were unable to coordinate together and so I decided to complete the rest of the work individually. My individual contributions to this project are:

- Implementing the android application as a whole which covers receiving sensor data over bluetooth, appending geo-location and timestamp information, connecting to firebase api and sending data to firestore database.
- Implementing the entire cloud based analytics pipeline using backend module (Python), visualization dashboard (Html, Javascript, Zing Charts) and the Notification System (Twilio API).
- Implementing Encryption-Decryption security feature using RSA algorithm.

Overall, I really enjoyed working on this project as it helped me to learn a lot. [1]

8 CONCLUSION

In this project, we have developed a full stack IoT based real time air quality monitoring system. With the use of NXP semiconductor development board and different sensors, we are able to collect the different pollutant concentrations in the air. We covered 4 different locations (indoor and outdoor) to collect pollutant concentrations and used a standard AQI formula to compute the air quality index of each location. Different analysis and visualizations have been performed on these collected measurements and many insightful observations have been made and depicted through the visualization dashboard. Other than this, a notification system has been implemented to alert the users subscribed to our monitoring service about the changes in the air quality at their location. The collected data and AQI readings have been verified against publicly available real time data for the city of Edinburgh. An asymmetric cryptography algorithm was used to secure the transmission of data from the embedded device to the Android Application. Overall, the system is efficient and is capable of capturing the trends in air quality in real time[2].

REFERENCES

[1] Air quality monitoring using iot and big data. https://www.gsma.com/iot/wp-content/uploads/2018/02/iot_clean_air_02_18.pdf.
[2] Sami Kaivonen and Edith C-H Ngai. Real-time air pollution monitoring with sensors on city bus. *Digital Communications and Networks*, 6(1):23–30, 2020.
[3] Frdm-k64f freedom module user's guide. https://www.mouser.com/catalog/specsheets/NXP_01112019_FRDM-K64F.pdf.
[4] Grove - multichannel gas senso. http://wiki.seeedstudio.com/Grove-Multichannel_Gas_Sensor/.
[5] Adafruit sgp30 tvoc/eco2 gas sensor. https://www.mouser.co.uk/datasheet/2/737/adafruit-sgp30-gas-tvoc-eco2-mox-sensor-1396564.pdf.
[6] Dust sensor user manual. https://www.waveshare.com/w/upload/0/0a/Dust-Sensor-User-Manual-EN.pdf.
[7] Adafruit bme280 i2c or spi temperature humidity pressure sensor. https://www.adafruit.com/product/2652.
[8] Hc-06 bluetooth module. https://components101.com/wireless/hc-06-bluetooth-module-pinout-datasheet.
[9] Get started with android. https://developer.android.com/.
[10] Get started with cloud firestore. https://firebase.google.com/docs/firestore/quickstart.