ISTANBUL TECHNICAL UNIVERSITY
FACULTY OF COMPUTER AND INFORMATICS

MOBILE AIR QUALITY MONITORING IN ITU CAMPUS

Graduation Project Final Report

Emre Onay
150140116

Department: Computer Engineering
Division: Computer Engineering

Advisor : Prof. Dr. Sema Fatma Oktuğ

May 2019

ISTANBUL TECHNICAL UNIVERSITY
FACULTY OF COMPUTER AND INFORMATICS

MOBILE AIR QUALITY MONITORING IN ITU CAMPUS

Graduation Project Final Report

Emre Onay
150140116

Department: Computer Engineering
Division: Computer Engineering

Advisor : Prof. Dr. Sema Fatma Oktuğ

May 2019

Statement of Authenticity

We hereby declare that in this study

1. all the content influenced from external references are cited clearly and in detail,
2. and all the remaining sections, especially the theoretical studies and implemented software/hardware that constitute the fundamental essence of this study is originated by our individual authenticity,

İstanbul, 25.05.2019

Emre Onay

AIR QUALITY MEASURING IOT APPLICATION IN ITU AYAZAGA CAMPUS (SUMMARY)

Today many universities around the world have various Smart Campus applications and some of these applications take atmospheric issues into account such as humidity, temperature, air pollution and so on. In this project, first, the air pollution in Ayazaga campus of Istanbul Technical University is measured. Then, a computer network is created which can take these measurements, transfer them, store them and present them. In addition to laying the foundations of Smart Campus applications in the name of Istanbul Technical University, the main aim of this project is to develop a mobile application which is used to monitor the air quality in the Istanbul Technical University Ayazaga campus and develop an environment to collect air quality data, transfer it through this environment, store it to the cloud database and be able to present it in various forms to the end user.

First step of this project is a sensor that can measure the air quality in the campus. In order to accomplish this measuring task, an economic but also reliable air quality sensor MQ-135 is chosen. MQ-135 has one positive, one negative voltage inputs and an analog signal output. First, it had to be calibrated for evaluating the right results. Calibration process was done with the help of an Arduino circuit. Some related programs were run on Arduino, the sensor was left outside for 12 hours until it obtains a stable value and the obtained value was saved into the running program on Arduino. After this point, thanks to this value and program, the sensor started to transmit the value it measures, uninterrupted.

The second node in the network is an Arduino UNO. As MQ-135 sensor needs a voltage source and can only send the computed data through analog signals, an Arduino needed to be added into the network. Its duty here is not only to transform the incoming analog data to rational “ppm” values with the help of its functions, but also to transfer them to the next node through the retrofitted Bluetooth module. For the sake of the project, Arduino UNO is chosen for this task, because it is easy to use, there are tons of resources on the Internet and once more it is an economic component. The 5V and Ground outputs on Arduino feed the air quality sensor and the analog input receives the incoming data. By this approach, the analog data is converted into “ppm” values and incorporated to the network. Ppm represents the particles per million. In this project, the parts that we have measured are the weighted average of the particles of the gasses which cause air pollution. The detailed explanation of this value is given inside the project. Calculated ppm values are sent to the Android application through Bluetooth. As Arduino UNO does not contain Bluetooth, an external HC-05 Bluetooth module is added to it. This module also gets its voltage inputs from Arduino and send/receive data through serial communication.

Android app is the another node in the network. It was written for only this project and has 2 modes named as user and collector. In this stage, the collector mode is active. The function of this mode is to create a Bluetooth connection with Arduino, give a meaning to the received data and send them to the cloud database. By appending location, date and time information, the data was given a meaning. Since the ITU Ayazaga campus is very big, it is needed to take plenty of measurements from plenty of different spots in the campus. These measurements also need to have location, date and time values so that they can have a characteristic when they reach to the end user. This procedure is done by taking location permission from the user when the

collector mode is opened. Collected data is pushed to the cloud database for storing and retrieving whenever they are needed.

MongoDB is a cloud database service with a free usage opportunity. Sticking by the location, date and time values, which are very important in this case, the data is stored in this database. Whenever a user makes a request, the relevant data is fetched from the cloud and brought back to the application for representation. This cloud database is the key feature of this multi node network. The end user, the user mode, is only in touch with this database, if there is a mistake or failure in this phase, it will cause serious problems in the application.

The last stage of the network and this project is the user mode of the Android application. In this mode, the data will be represented to the user. There are two ways of representation. First one is colouring on the ITU Ayazaga campus map. The campus is divided into multiple zones and each node is being coloured according to its own air quality value. 5 different colours are chosen with respect to the quality measures. These colours are, from the cleanest to the dirtiest air, light green, dark green, yellow, orange and red. The limits and endpoints of these categories are denoted inside the report. Second representation method is graphs. The user can see the air quality values of the region he/she chooses from the graphs. In this step, there are various choices. Daily, weekly and monthly values can be displayed. Moreover, for each day, there are three options as morning, afternoon and night. All these data are collected by authorized collectors taking measurements from different spots in the campus. This method is chosen because of the limited budget. Since this is not a very sustainable way, it was not possible to collect data every day. However, there are still enough data for representation and for successful operation of the application.

İTÜ AYAZAĞA KAMPÜSÜNDE HAVA KİRLİLİĞİ ÖLÇÜMÜ
(ÖZET)

Günümüzde dünyanın birçok üniversitesinde çeşitli akıllı kampüs uygulamaları bulunuyor ve bu uygulamaların bazıları kampüs içerisindeki nem, sıcaklık, kirlilik gibi atmosferik verilere dayanıyor. Bu projede İstanbul Teknik Üniversitesi Ayazağa kampüsündeki hava kirliği ölçülmüş ve bu ölçüm verilerini alıp saklayan, aktaran, depolayan ve görüntüleyebilen bir bilgisayar ağı kurulmuştur. Buradaki asıl amaç İstanbul Teknik Üniversitesi adına akıllı kampüs çalışmalarına bir giriş niteliği taşımanın yanında aslen kampüs içerisindeki hava kalitesinin görüntülenebileceği bir mobil uygulama geliştirmek ve bu uygulamanın çalışabileceği bir ortam yaratmaktır. Bu ortam da güvenilir, ölçeklenebilir, çok öğeli bir bilgisayar ağı yaratmak, bu ağ üzerinden veri transferi yapabilmek, bilgileri depolayabilmek ve bunları anlamlandırıp son kullanıcıya pratik bir şekilde sunabilmek üzerine kuruludur.

Bu projenin ilk aşaması kampüsün içindeki hava kirliliğini ölçebilecek bir sensördür. Bu görevi yerine getirebilmesi için hesaplı fakat güvenilir bir hava kirliliği sensörü olan MQ-135 kullanılmıştır. MQ-135 bir pozitif bir negatif uca sahip olan ve analog çıkış veren bir sensördür. Bulunduğu ortamdaki kirliliği doğru bir şekilde ölçebilmesi için önce kalibre edilmiştir. Kalibrasyon işlemi, bir Arduino devresi yardımıyla yapılmıştır. Arduino üzerinde çeşitli fonksiyonlar çalıştırılmış, sensör 12 saat açık alanda bırakılarak sabit bir değer alması beklenmiş ve değer Arduino üzerinde çalışan fonksiyonlara kaydedilmiştir. Bu noktadan sonra yine bu değerin ve fonksiyonların yardımıyla sensör kesintisiz olarak ölçümünü yaptığı değeri aktarmaya başlamıştır.

Bilgisayar ağının ikinci elemanı bir adet Arduino UNO'dur. MQ-135 sensörü bir voltaj beslemesine ihtiyaç duyduğu ve ölçtüğü verileri sadece analog sinyaller şeklinde iletebildiği için Arduino'nun bu ağa dahil edilmesi gerekmiştir. Burdaki görevi, sensörden gelen analog değeri fonksiyonlarını kullanarak mantıklı bir "ppm" değerine dönüştürmek ve üzerine eklenmiş bulunan Bluetooth modülü yardımıyla ağın bir sonraki elemanına aktarmaktır. Bu iş için hem kullanımı kolay olması hem internet üzerinde hakkında birçok kaynak bulunması hem de hesaplı bir ürün olması nedeniyle Arduino UNO seçilmiştir. Arduino, üzerinde bulunan 5V ve Ground çıkışları ile sensörü beslemektedir ve analog girişi sayesinde gönderilen verileri alabilmektedir. Analog değeri "ppm" değeri haline getirip veriyi anlamlandırmış ve ağa dahil etmiş olur. Ppm, parts per million, yani milyondaki tanecik sayısını temsil eder. Bu projede ölçümünü yaptığımız parçacıklar hava kirliliği yaratan gazların atmosfere saldıkları parçacıkların ağırlıklı toplamıdır. Değerin detaylı açıklaması raporun içinde verilmiştir. Hesaplanan ppm değerleri bluetooth aracılığıyla Android uygulamasına gönderilir. Arduino UNO halihazırda bir Bluetooth özelliği bulundurmadığı için kendisine bir HC-05 Bluetooth modülü eklenmiştir. Bu modül de beslemelerini Arduinodan sağlamakta ve seri haberleşme ile veri alıp vermektedir.

Android uygulaması ağdaki bir diğer elemandır. Bu proje için özel yazılmış olup kullanıcı ve toplayıcı olmak üzere iki moda sahiptir. Bu aşamada toplayıcı modu aktiftir. Modun görevi Arduino ile bir Bluetooth bağlantısı kurmak, aldığı verileri anlamlandırmak ve bulut veritabanına yollamaktır. Verileri anlamlandırmaktan kasıt ölçülen değere lokasyon ve tarih, saat bilgisi katmaktır. İTÜ Ayazağa kampüsünün oldukça büyük olması nedeniyle birçok

noktasından birçok ölçüm yapılmalıdır ve bu ölçümlerin başarılı bir şekilde son kullanıcı ile buluşabilmesi için alınan değerlere tarih,saat ve lokasyon bilgisi eklenmelidir. Uygulama bu işlemi, toplayıcı modu açılırken kullanıcıdan lokasyon bilgisi iznini alarak yapar. Toplanan veriler, depolanmak ve isteğe göre çekilip görüntülenmek amacıyla bir bulut veritabanına aktarılır.

MongoDB ücretsiz sürüme sahip olan bir bulut veritabanıdır. Alınan veriler tarih, saat ve lokasyon bilgilerine sadık kalınarak bu veritabanında depolanır. Kullanıcı bir istek yaptığında veriler burdan alınır ve ekrana yansıtılır. Çok öğeli bilgisayar ağının önemli bir parçasıdır. Son kullanıcı sadece bu veritabanıyla bağlantıdadır ve eğer veritabanı çökerse veya bilgilerde kayıp yaşanırsa uygulamada sorun yaşanacaktır.

Projenin ve ağın son aşaması Android uygulamasının kullanıcı modudur. Bu modda bilgiler kullanıcıya sergilenecektir. İki tur sunum şekli bulundurur. Birincisi İTÜ Ayazağa Kampüsü haritası üzerindeki renklendirmedir. Kampüs birçok bölgeye ayrılmıştır ve her bir bölge hava kirliliği değerine bağlı olarak renklendirilir. Hava kirliliğinin değeri üzerinden 5 renklendirme yapılmıştır. Bunlar en temizden en kirliye doğru olmak üzere açık yeşil, koyu yeşil, sarı, turuncu ve kırmızıdır. Bu renklerin sınırları rapor içerisinde belirtilmiştir. İkinci sunum şekliyse grafiklerdir. Kullanıcı seçtiği bölgenin kirlilik değerlerini grafikler üzerinden görebilir. Bu aşamada da çeşitli seçenekler mevcuttur. Günlük, haftalık veya aylık değerler seçilebilir. Ayrıca her gün içinde de sabah, öğleden sonra ve akşam olmak üzere 3 ayrı ölçüm bulunur. Tüm bu veriler, yetkili toplayıcıların sensörle birlikte kampüs içerisindeki noktalardan ölçüm almalarıyla toplanmıştır. Kısıtlı bütçeler nedeniyle bu yöntem seçilmiştir. Yöntemin zorlukları nedeniyle her gün ve her saate dair değerler elde edilememiştir fakat uygulamanın başarılı bir şekilde çalışması ve kullanıcı tarafından gözlemlenebilmesi için gerekli sayıda veri mevcuttur.

Contents

1 Introduction and Project Summary

Internet of Things, IoT, is one of the fastest growing concepts of the technology. Nowadays it is getting even bigger investments and new projects are being developed. Essentially, it aims to make everyday objects, things, become online by connecting to the Internet. These online things become remotely controllable and they gain the ability to freely communicate with each other or with the mobile phones/PCs people use every day. First step of IoT is measuring physical parameters of the world and the most basic elements of this operation are the sensors. These sensors act like tentacles of the networks, they sense the desired data and send incorporate them to the network. One of the most common used types of these sensors are the sensors with capability of sensing the atmospheric factors. These factors can be air temperature, humidity or pressure as well as the air quality. Air quality is the focus of this project.

Air quality is found according to the portions of pollutant gasses and particles in the air. Namely, these pollutant gasses are carbon monoxide, nitrogen oxides, sulphur oxides and ozone. In addition to these gasses, also particles smaller than 10 micrometres and 2.5 micrometres are taken to the account. There is not a certain way to find the weighted air quality. Each country assigns different weights to different gasses, but all of them use the same gasses for evaluation. Every gas's proportion in the atmosphere is denoted as "ppm" values. Ppm means parts per million, it indicates how many particles out of one million particles belong to the desired gas in the atmosphere. These elements do not always stay stable in the atmosphere, the levels of these elements can change in every day and even in every part of the day. MQ-135 is a low cost air quality sensor for sensing and measuring the gasses in the atmosphere and calculating an average for them.

The aim of this project is to develop a mobile application which is used to monitor the air quality in the Istanbul Technical University Ayazaga campus and develop an environment to collect air quality data, transfer it through this environment, store it to the cloud database and be able to present it in various forms to the end user. Taking measurements is also important, but the actual focus in this project is on successfully implementing and realizing the network. This environment contains five components; air quality sensor, Arduino board, collector mode of Android application, cloud database and user mode of Android application. A simple representation of the environment can be seen in Fig. 1.1.

Firstly, the air quality in various points of the campus needs to be measured. Measuring the values in just one time is not enough to find out the desired quality, so these operations need to be repeated in fixed periods. For this operations, MQ-135 air quality sensor is used. It is a fast response sensor with high sensitivity, wide scope of detecting gasses and a stable, long life. This sensor is not fully operable as it is manufactured. Before expecting reliable values from it, it has to be calibrated. This calibration is done with respect to a gas's fixed proportion in the air. For example, carbon dioxide has a somehow fixed value in the atmosphere, it does not change its presence very frequently. So, with the help of an Arduino program, the fixed ppm value of carbon dioxide is entered to the program and by leaving the sensor outside for 12 or more hours, the sensor gets calibrated and gets ready to work. It operates with 5 volts and outputs the measured value as analog signal.

As mentioned above, an Arduino UNO board is used for taking values from the sensor. Arduino UNO is a microcontroller development board with open-source implementation and libraries. It is easy to program and has low cost which makes it suitable for this project. It has two fields

for programming, setup() and loop(). Setup() runs for once, then loop() runs forever. Before the setup(), the MQ-135 libraries for calibration and measuring are added. In the setup part, the ports for serial communication is determined. In the loop part, the gas sensor variable is set to analog port 0, because the sensor sends the data in analog signals. Then, it checks the serial communication, which is Bluetooth connection here. If it is available, in 5 seconds periods, Arduino reads the value from its analog input and sends it through Bluetooth to the Android application.

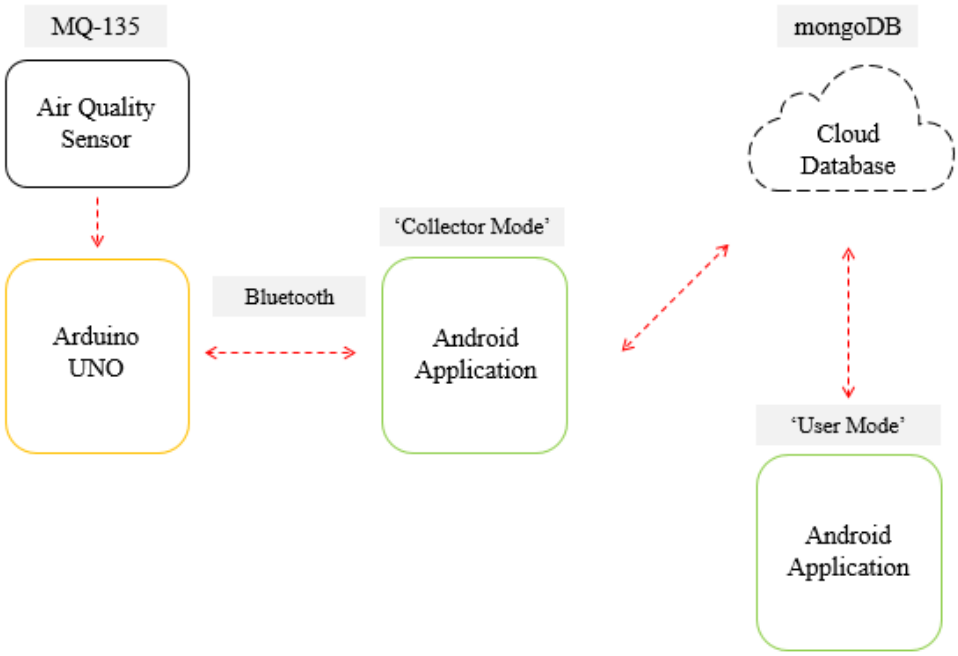


Fig 1.1: The representation of the environment

The gateway for MQ-135 and Arduino UNO is the Android application. This app has 2 major tasks. One is for end users, it provides and represents the requested data to the user. Second is for collectors, it connects to the Arduino, receive data from it and sends it to the cloud database. Collectors need to carry the sensor and the Arduino with themselves for making measurements around the campus. As there are not enough sensors for fixing to the various points in campus, this data collecting operations need to be done manually. Authorized collectors enter to the collector section with a password. This section is another activity in the app, when entered, it requests location permission from the collector and asks for opening the location and the Bluetooth so that it can connect to the Arduino. The collector pushes the "connect" button and the app searches for Arduino's Bluetooth module. If it is found, it connects directly. After the connection, the ppm values measured by the sensor start to flow on the screen with 5 seconds intervals. First, the values are inconsistent because the sensor is just brought to that point. After 1-2 minutes the values settle down and the collector pushes the "send data" button. A dialog box appears and lets the collector check the values: longitude, latitude and the air quality value. If everything is okay, the box is confirmed. After the confirmation, the app adds important time and date information to the air quality value and sends it to database. As the user is able to choose date, time and location to display, the data need to be pushed to the database with this information.

The database in this project is a cloud database, MongoDB. It is a non-relational database which means the incoming data has the form of a document, not a table, and data with different variables can be hold together. MongoDB provides a 50 GB free database space for its users, so this will be suitable for this low cost project. The air quality data only has date, time, location and quality value variables, this does not take much space, 50 GB are more than enough for this project. Moreover, as there is only one use case, rather than using a MySQL server, this simple interfaced cloud database is more practical. All data coming from all spots in the campus and at all times of the day will be kept in this cloud database. The programming part is for connecting the database, sending data and receiving data according to the need.

The user part of the Android application forms the last part of the network and the project. User mode is for the end user and it displays wanted data to the screen. This representation is done in two ways. First way is on an ITU campus map with coloration. Map is divided into zones and each zone has a colour according to its air quality. The zones' average air quality values are calculated by taking every collection spots to account inside their region. This value can be in the range of 5 colours: light green, dark green, yellow, orange and red. This order is from the cleanest to the dirtiest air. Second way of representation is graphs. The user can pick a region and see its air quality results for specific time range on the graphs. There are also daily, weekly and monthly choices which affects the graphs. The user can also choose a date from the calendar and see the related values on the map and on the graphs. The needed data are fetched from the database with given filters and brought to the application for display. Previously measured values are kept in the database for backward compatibility of future researches.

2 Literature Survey

As Internet of Things is spreading wide around every corner possible, the need of well-detailed explanation of IoT architecture for every developer who is interested in launching out a product increased dramatically. Latterly, in 2015, Al-Fuqaha et al published a great paper explaining IoT and IoT applications from their very basics to architectural design principles while emphasizing significances of IoT technologies, communication infrastructure and encountered problems [1]. They divided IoT elements in six section to give a better view of design and application in every segment. This project is planted on those elements. The air quality sensor will be the eyes and ears of the system providing it sensing capability. As for communication, Bluetooth protocol, which is incorporated to the Arduino through an external module, will be responsible to send and receive data with other devices. Since the sensor device does not have high computational power and storing capabilities, another service will be in charge of making statistical calculations. This service will be the Arduino UNO, it will be a bridge and provide the necessary interface between the sensor and the other nodes.

The advantages of IoT devices over conventional micro controllers and computers for extensible work designs has been stated in a whitepaper in 2014 by Freescale and ARM, two very important company in semiconductor production area [2]. While lowering component prices and electrical costs, increasing scalability, dependability and features as well as allowing mixed systems to be used together easily and securely, IoT environment becomes the smartest choice for projects like this that needs many scattered, small and efficient devices to work with. The whitepaper also compares communication technologies and suggests usage for them. As for the BLE protocol, range is up to 10 meters with transfer rate of up to 1Mbps while it is still stated being highly energy efficient. The Bluetooth module used in this project is HC-05 which is fully qualified V2.0+EDR (Enhanced Data Rate) device. It has 3Mbps modulation with complete 2.4GHz radio transceiver and baseband. These values are suitable and sufficient in the scope of this project. It is connected to the Arduino UNO and provides connection and data transmission between the Arduino and the Android application.

In a recent, very detailed, IoT supported, air quality involved project, Chen et al. proposes an air quality scale called Multidimensional Air Quality Indicator (M-AOI) to rate the impact of pollution in air to human health as respiratory diseases and psychological problems. In the project, they use IoT sensing in different ways such as by smart clothing, car and buildings as well as smart phone applications and getting meteorological data directly. The whole data is stored in a way to form a big data system (M-AQI big data) and used by big data processing services on supercomputers to show the relations between diseases map and air quality map. As a next step, based on the results the user may be suggested to apply a healthcare plan [3]. Their large-scale project is important to show that IoT has already started to play a big role to gather and transfer data to analyse and make inferences. For this project, it proves IoT crowd sourcing and sensing is an applicable way to gather air quality data in an area. And it also provides valuable remarks in how air quality can be rated based on different particles quantity in air.

Published on April of 2018 in a project-sharing web site, Sharma explains a hardware project to measure and show present air quality using very low cost hardware components in do-it-yourself way of implementation. This project works on an Arduino platform and uses an MQ type MQ-135 air quality measuring sensor. Although this project does not fully contain the

parts that are used in Sharma's project, still the UART interface is used to connect a device by their developer kits [4]. Therefore, this project in particular shows that a low cost sensor can be used in a way to measure air quality data and they can connect to various other sensors or microcontroller devices by UART interface.

The air quality sensor MQ-135 has an unrevealing datasheet. The perfect explanation of how it works is not present in the paper. So, most of the users and researches have tried to uncover the characteristics of this sensor by using the graphs on the datasheet. Davide Gironi was the primary actor who has made inferences by applying mathematical equations such as power series applications on the graphs [5]. In this blog post, he does correlation function estimations of MQ-135 gas sensor and comes up with a way to make the sensor fully functional by calibrating it. Gironi's works are approved and reused by other project developers later on. He says that based on the somehow fixed amount of carbon dioxide gas in the atmosphere, the sensor can be calibrated to find the air pollutants' weighted average ppm in the atmosphere. This is done by connecting the sensor to the Arduino, running the specified program on it and leaving the MQ-135 outside for 12 hours to let it find a stable value. This was also chosen as the way for calibrating the sensor in this project. After this step, it was possible to get logical, uninterrupted inputs from the sensor to the Arduino.

One of the most complete and notable realizations of Smart Campus on Air Quality Monitoring belongs to Ali, Soe and Weller, 2015 [6]. Their goal was to monitor ambient air quality in school surroundings, they chose a school for the experiment and created a real time, ZigBee-based, wireless, low-cost air quality measuring system. Just as in this project, the sensor was MQ-135 and the connected node which was used for taking and transmitting the data of the sensor was Arduino UNO. As there are various ways of implementing a network, they chose to create a ZigBee-based WSN. Moreover, a solar panel was used for providing energy and LabVIEW tool for presenting the results. However, this project aimed to make things simpler, so Arduino uses a Bluetooth module for connecting to the Android application and the application acts as a gateway and as a display.

The World Air Quality Index project is a non-profit project started in 2007, located in China. Having more than 11000 stations, the project measure and provide air quality information to 88 countries. Data are obtained from each country by respective Environmental Protection Agency (EPA). In one of the posts of World Air Quality Index project, it is stated that, for monitoring the air quality in one place, 24 hours averaging method does not make sense. Since one strong wind can clean the air or one big fire can pollute it badly in small amount of time, 24 hours of measuring and then interpreting the quality results is a bad idea. Rather than using this method, momentary monitoring, named as NowCast, is much more logical and gives more accurate values [7]. As the same way it is stated in this post, the measurements within the scope of this project is also done momentarily. The user is able to choose the day and the time for monitoring the air quality results.

3 Developed Approach and System Model

This project is developed for measuring the air quality in the Istanbul Technical University Ayazaga campus and creating a network for transmitting, storing and presenting the measured data. This network has 5 nodes: the air quality sensor, Arduino UNO board, Android application in collector mode, MongoDB cloud database and Android application in user mode.

These 5 nodes create a chain network, which means a node can only reach to and communicate with its one or two consecutive nodes in the network. The connections are between the air quality sensor and the Arduino, the Arduino and the Android application's collector mode, the collector mode and the cloud database, the cloud database and the user mode of the application.

The first two components of the network, which are the sensor and the Arduino, are offline. There is a physical connection between these two. Arduino uses the Android app as a gateway to the Internet. The data transfer here is done by a Bluetooth connection. The rest of the network, collector mode, cloud database and user mode, is online. The app sends data to and receives data from the cloud database, MongoDB, through the internet. The database is the storage point of the network and the user mode of the Android application is the presentation section.

3.1 Data Model

The data in this project can be examined in 3 different versions. First one is the analog signal output of the sensor, the second one is the ppm value from Arduino to the collector mode and the third one is enriched data from the collector mode to the cloud database. This version of the data is also fetched from the database and used while representing the data on the map and the graph.

3.1.1 Analog Signal Data

There are 2 types of outputs that can be taken from the air quality sensor, MQ-135. These are analog and digital outputs. The digital output works as follows: there is an adjustable switch on the circuit of the sensor. That switch can be physically set to a value. When the air quality breaches that value, the sensor gives a digital signal, 1.

However, this is not the desired output. As a more precise and nonstop data flow is wanted, the analog output is used. MQ-135 sends the data it measured continuously from its analog output to the Arduino board.

3.1.2 PPM Value

Arduino board receives the analog signal from its input, but the incoming data is not the exact ppm value. The MQ-135 library which is added to the Arduino program helps Arduino to convert the incoming data to a ppm value. As mentioned before, ppm means parts per million, it represents the proportion of the pollutants gasses' particles over 1 million particles in the atmosphere.

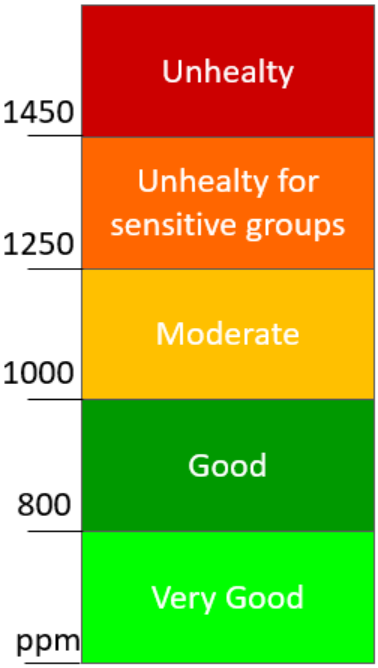


Fig 3.1.2: Air Quality grading chart

Based on the measurements, which were done in different times and in different days, the ppm value range in different zones of ITU Campus was found between 600 – 1500 ppm. According to these values, as it can be seen in Fig. 3.1.2, a unique air quality chart is created for the campus. These are the colours that can be seen on the map in the application in the presentation section. After finding the ppm value, it is sent to the Android app through Bluetooth.

3.1.3 Enriched Data

Android app connects to the Arduino by Bluetooth. Within 5 seconds periods, Arduino sends the message "Air quality is:XX:ppm.". The app reads the string from the input stream and stores into a string. Then, the string is decomposed with respect to ":" character, the air quality is obtained and converted to double value.

After obtaining, every ppm value needs to be given a characteristic. This characteristic can be described as date, time and location data. Unless a value has these data, it is not useful and will be lost in the database.

```
Document canvas = new Document("month", month)
        .append("week", week)
        .append("day", day)
        .append("hour", hour)
        .append("lat", latitude)
        .append("long", longitude)
        .append("value", ppm);
```

Fig 3.1.1.1: Creating the enriched data

In Fig. 3.1.1.1, the creation of the enriched that is shown. This operation is done inside the collector mode of the Android application. A document is created for each ppm value and it is pushed to the database.

MongoDB is used as a cloud database. MongoDB is a non-relational database. This means the data is uploaded to the database in form of documents, not in form of table entries. It can be seen in Fig. 3.1.1.1. This brings the advantage of inserting different documents to the same database. Non-relational databases are also useful when the data does not have certain attributes all the time. In relational databases, the admin must first create a table with certain entry attributes like day, time, etc. Only entries with these attributes are accepted to these databases. However, in MongoDB, this is not the case, there are no tables. In this project, the data is pushed with date, time and location data, but in the future, some new features can be added to the new measurements and all of this documents can still be in the same database, the user can still operate on these data together. In Fig. 3.1.1.2, there is a picture of one entry in the MongoDB database. It indicates, the ppm value at (41.10318387, 29.0283657) coordinates on 14th of May (1 is added to the month value) at 8 pm was 809,73.


```
_id: ObjectId("5cdaf7c55bb1d10b0ed0c4fd")
month: 4
week: 3
day: 14
hour: 8
lat: 41.10318387
long: 29.0283657
value: 809.73
```


Fig 3.1.1.2: Representation of a document in the database

3.2 Structural Model

In this part, the structural model of the network is shown and explained. Moreover, the inner structure of the android app is described. Fig. 3.2.1 gives an overview of the system.

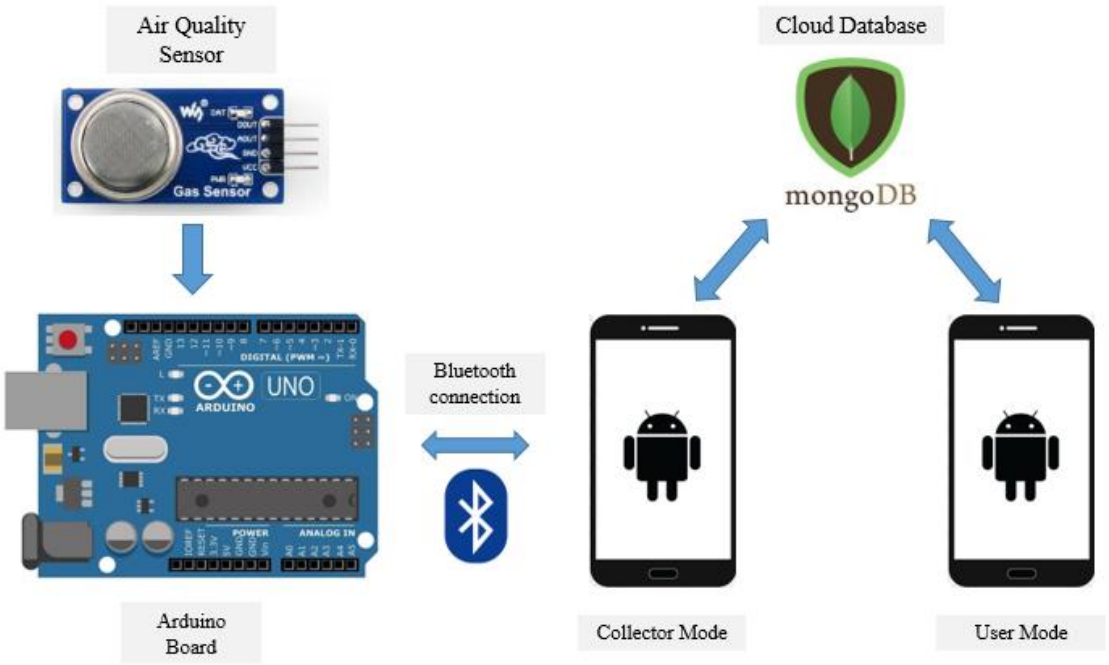


Fig 3.2.1: The representation of the network

The air quality sensor is physically connected to the Arduino. It sends analog signals from that physical cable connection, but does not get any data from Arduino. Hence, it is a one-way communication. The connection of these two components can be seen in Fig. 3.2.2. Arduino feeds MQ-135 with 5 Volts and Ground, the analog output of the sensor is connected to the A0 (Analog 0) input of the Arduino UNO board. Before the usage, MQ-135 needs to be heated for 5 to 10 minutes.

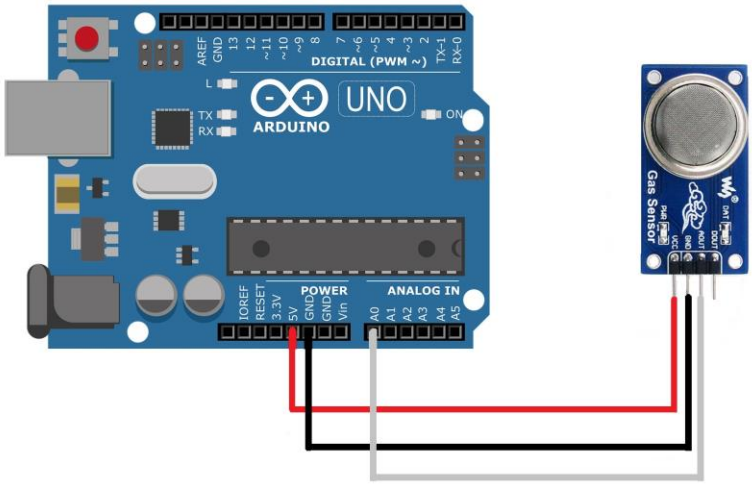


Fig 3.2.2: Hardware design of Arduino and MQ-135 [8]

The second connection is between the Arduino and the collector mode of the application. When the user wants to switch to the collector mode, the app asks for a password. If password is accepted, the collector activity begins. In order to use this mode, the location permission needs to be given to the app. The app opens the location service and the Bluetooth service of the phone. This mode has a simple interface. When the collector pushes “Connect”, the app looks for the Bluetooth module of the Arduino board and connects to it. This connection is done by creating an insecure RF communication socket by using the UUID value of "00001101-0000-1000-8000-00805F9B34FB". After the connection, in 5 seconds periods, the calculated ppm value starts to flow on the screen. Arduino circuit has a very simple loop code for this operation, given in Fig 3.2.3, When the values settle, collector pushes to the “upload data” button and the data is added to the database with date, time and location attributes. This collector mode screen and the discussed activities can be seen in the Fig. 3.2.4.

```
Loop
        Define gas sensor to A0
        While connected to Bluetooth
                quality = sensor.getPPM()
                Serial.print "Air quality is:quality:"
                wait 5 seconds
        end while
end loop
```

Fig 3.2.3: The pseudocode of the Arduino program

Fig 3.2.4: Collector Mode screen of the application

The next connection in the environment is the connection between the collector mode and the cloud database. The collector mode has an asynchronous task, which is named as ConnectBT. This ConnectBT extends AsyncTask class and provides connection to Arduino's Bluetooth module in the background. This class is the provider of the ppm values. In 5 seconds periods, it reads the value from the input stream, updates the ppm variable of the activity and then appends this value to the text view on the screen. There is another asynchronous task in Collector Activity and it is used for establishing the connection to the MongoDB. This class is MongoData and it also extends AsyncTask class in Java. As described above, it is called by the collector by pushing to the "upload data" button. It is called with some parameters. These parameters are longitude, latitude and the ppm value. Then, MongoData connects to MongoDB, gets the database and the collection inside the database, it creates the document, loads a characteristic to it as described above and inserts the data to the collection. Pseudocode of this operation can be seen in Fig. 3.2.5.

```
uri = MongoClientURI(the uri of the database)
mongoClient = MongoClient(uri)
database1 = mongoClient.getDatabase("database1")
collection1 = database1.getCollection("collection1")

create document
append details to the document
collection1.insert(document)
```

Fig 3.2.5: Pushing a document to MongoDB

Last interaction inside the environment is between MongoDB cloud database and the user mode of the Android application. The user mode presents the requested data to the user. This data can have specifications as date, time of the day, zone and time interval. This time interval can be

daily, weekly or monthly. Time of the day can be morning, afternoon or night. Date can be any date in which the application has some data to display. Finally, zone can be the whole of the campus or just one zone can be chosen by clicking on the respective rectangle. Average of the air quality data taken from inside of their borders defines the colour of the zones. Fig. 3.2.6 shows the pseudocode of how a zone gets coloured. Borders of the zones were defined when they were initialized. So, the algorithm fetches the evaluation points inside the zones' borders from the database, finds the average ppm value of those points and according to that value, it recolours the zone with respect to the chart in Fig. 3.1.2.

```
coll := collection from the database
for each zone in zones
        read borders to x1 x2 y1 y2
        points := collection.find(greaterthan("longitude", x1)) &&
        collection.find(smallerthan("longitude", x2))           &&
        collection.find(greaterthan("latitude", y1))            &&
        collection.find(smallerthan("latitude", y2))
        avg := getAverage(points.ppm)
        zone.setColour(avg)
end for
```

Fig 3.2.6: Pseudocode for colouring a zone

In Fig. 3.2.7, a representative user mode screen can be seen. There is a calendar icon on the top right for choosing the date. There are 3 tabs for daily, weekly and monthly projections. The main component on the screen is the ITU Ayazaga campus map on which the air quality monitoring is done. Zones are coloured according to the air quality they contain. Below the map there are 3 more tabs for morning, afternoon and night selections. Morning covers the time between 8.00-13.00, afternoon between 13.00-18.00 and night between 18.00-00.00. The last element of the page is a scroll view consists of graphical representations. This graphs relates and shows the previous and the current data. According to the time interval choices the date range of this graphs can vary.

There are 5 different colours in Fig. 3.2.7, just as it was presented in data model section, in Fig. 3.1.2. The most air pollutant zones are the ones near the road, near the entrance of the campus. Rectangles with the best air quality reside around the "golet". There are still areas with grey colour, this means no measurements are taken from those zones. Under the map, 3 graphs can be seen in the scroll view. First map shows the ppm values of pollutant gasses in the chosen zone on the chosen date for morning, afternoon and night. The second graph does the same thing for one day before the chosen date, the third graph does the same for two days before. Third graph can be seen by scrolling down the graphs. When switched to weekly or monthly mode, graphs also switch themselves to weekly and monthly statuses.

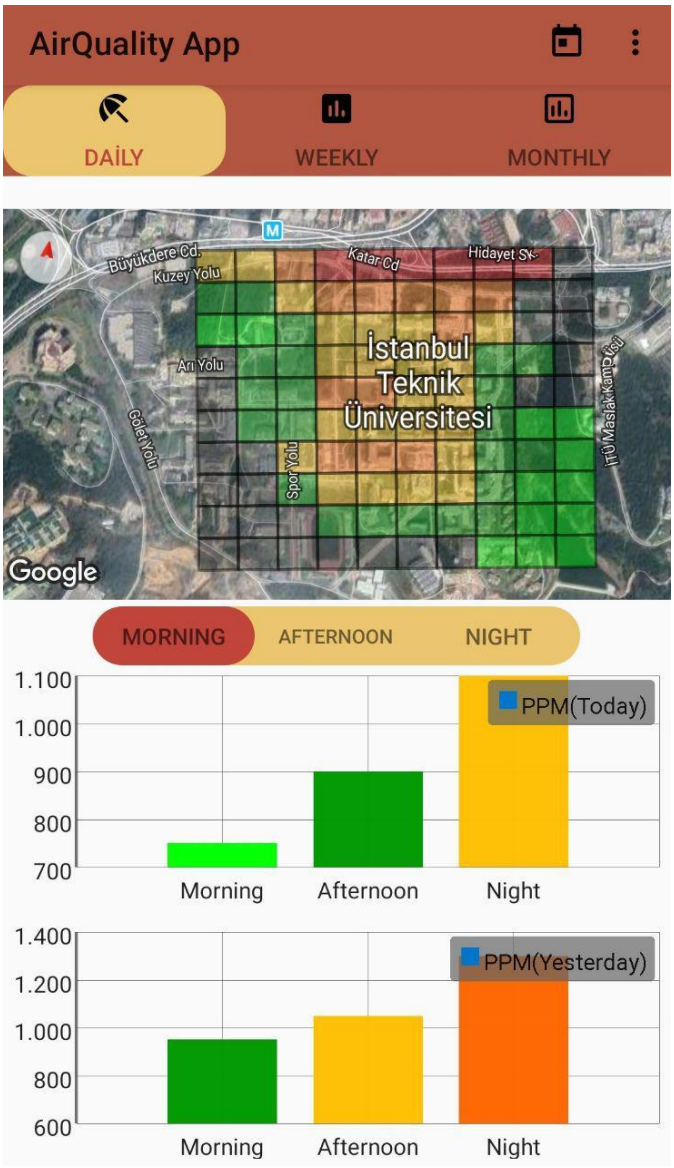
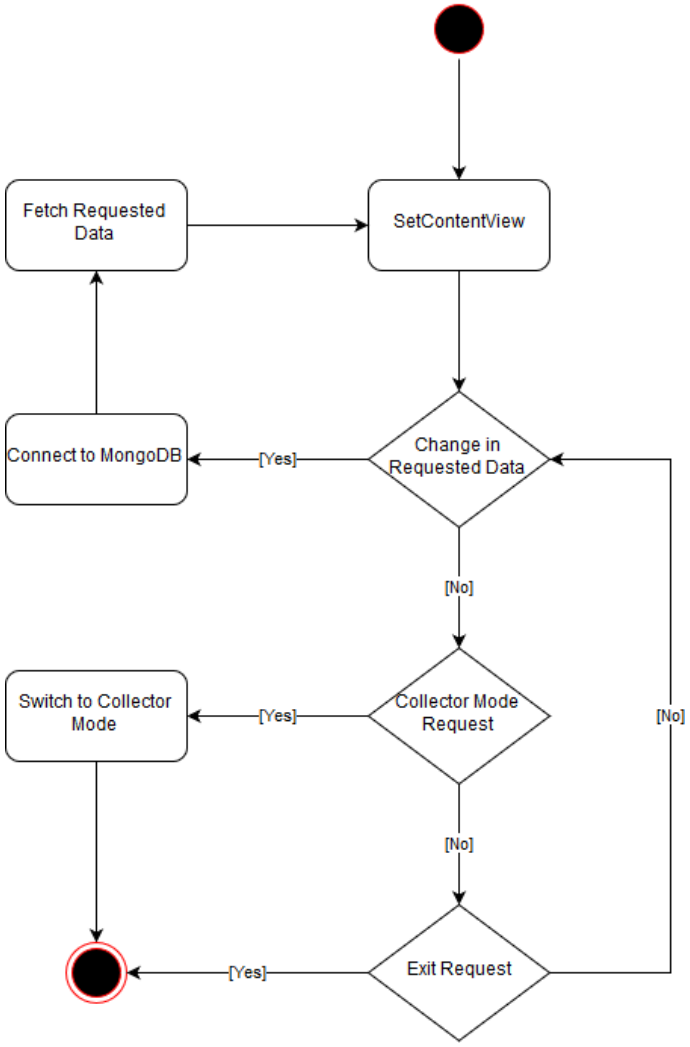


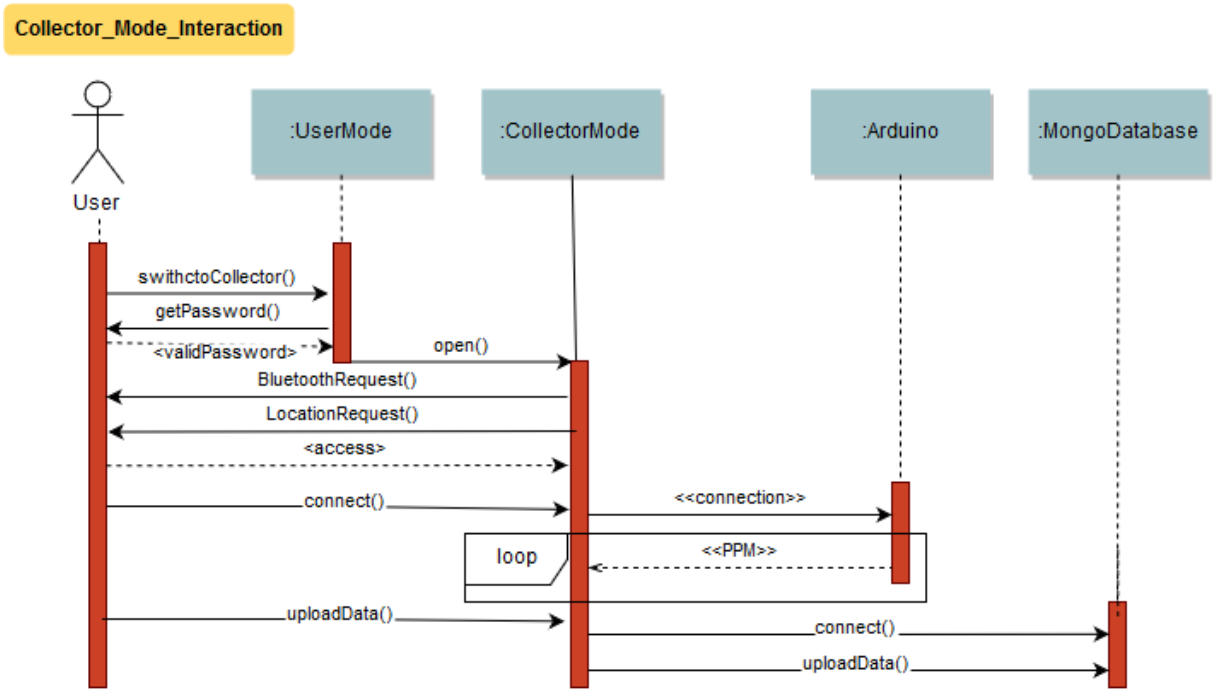
Fig 3.2.7: Example working screenshot from the application

3.3 Dynamic Model

3.3.1 User Mode State Diagram



3.3.2 Collector Mode Sequence Diagram



4 Experimentation Environment and Experiment Design

This project does not contain precise integration tests or some experimentations with data sets. However, a mobile application for air quality monitoring is developed in this project and an environment, including the application, is created from scratch. So, it is appropriate to describe this environment in this section. Moreover, collection methods of the air quality data, transfer, store and presentation operations of this data over the created network are also represented.

The environment of this project consists of 5 elements: air quality sensor, Arduino board, Android application's collector mode, cloud database and Android application's user mode. The place, the application area, where the data are collected from is ITU Ayazaga campus. Due to the budget issues, there is only 1 sensor and 1 Arduino. So, the data collection phase of the project is done manually. If there was enough budget, plenty of sensors and Arduino's could be placed inside the various spots in the campus, but then the scope of the project would have exceeded being a graduation project.

Fig 4.1: ITU Ayazaga Campus

In Fig. 4.1, the satellite view of the Ayazaga campus can be seen. For monitoring the air quality, the measured data must somehow be reflected to this map view. There could be a few choices at this point. A heat map representation was one of those choices, but this method would have required a lot more measurements and different tools. The appropriate display method was a grid view. Each rectangle in the grid can fetch its relevant data from the cloud database and colour itself according to the average ppm value of the data it has fetched. Relevant data means the data which are located inside the borders of the rectangle. By this method, a grid view can operate with less measurements. The realization of this grid view method is shown in Fig. 4.2. This grid view operation is not done in the Photoshop; each rectangle is a Polygon which is placed onto the map view in Android Studio. First, a start zone is chosen and its coordinates are found by investigating Google Maps. After creating one rectangle, all other rectangles are created consecutively. Each rectangle's coordinates are found with respect to the previously created rectangle. This code with coordinates generation is done by a simple program in Visual Studio. The created Java code is written to a text file. Then, the code is taken from the text file

and pasted to Android Studio. Initially, all rectangles have a grey colour, because they haven't get any data from the cloud database, yet.

Fig 4.2: Grid view of ITU Ayazaga Campus

As mentioned above, the measurements are done manually in this project. Few people were authorized as collectors. They had the password to the collector mode and they had coupled their Android devices with Arduino's Bluetooth module for easier connection. The collectors need to carry the sensor, the Arduino and the power source of the Arduino with them. This power source is mostly a laptop; it can also be a long-life battery between 7-12 Volts. Bluetooth and location services must also be opened; the application asks for these anyway. The collectors try to reach every part of the campus, but this is a tiring process, so it is not possible to collect data from every spot of the campus by this manual method.

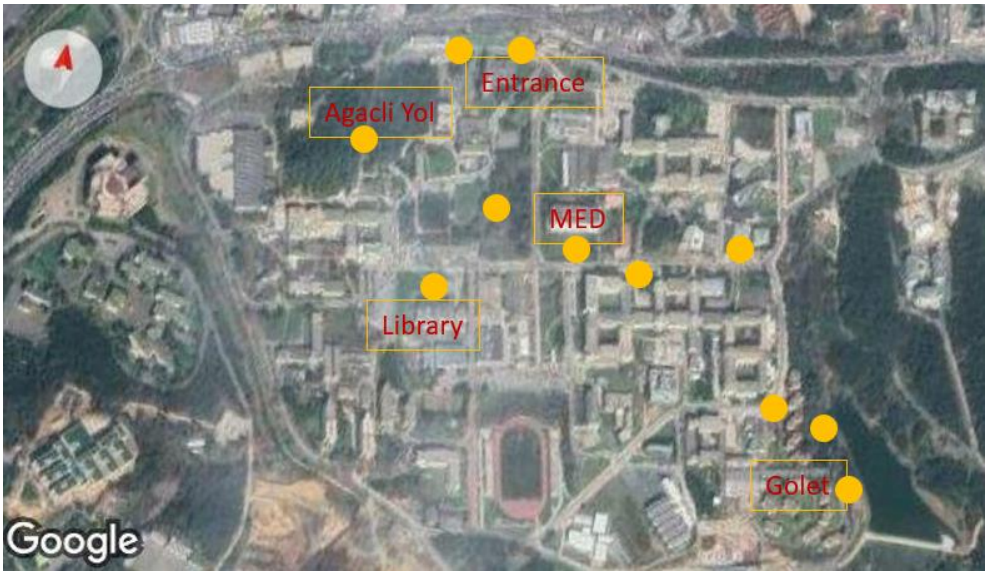


Fig 4.3: Marked measuring spots

Details of how to collect data is already described above, see part 3.2. This process needs to be repeated as many days as it can be and as many different hours it can be. Hence, in order to decrease the effort that must be put in for taking measurements, a few more important

measurement spots are determined and reported to the collectors. After that time, the data collection process was concentrated on these locations. In Fig. 4.3. these spots are marked on the map. There were reasons for choosing these sections. For example, the main entrance of the campus is a candidate region for the lowest air quality region, because from just outside that entrance, ten thousands of motor vehicles passes every day. The “agacli yol” and “golet” are strong nominees for having the best air quality in the campus because no vehicles pass near this long road and big lake. Moreover, the outside of “MED” is an interesting place to measure. That location has the densest undergrad student population in every hour of the day. In addition to these, the road from the Electrical Engineering faculty to the Chemical Engineering faculty is chosen, because most of the vehicle traffic inside the campus is placed there, including the ring services.

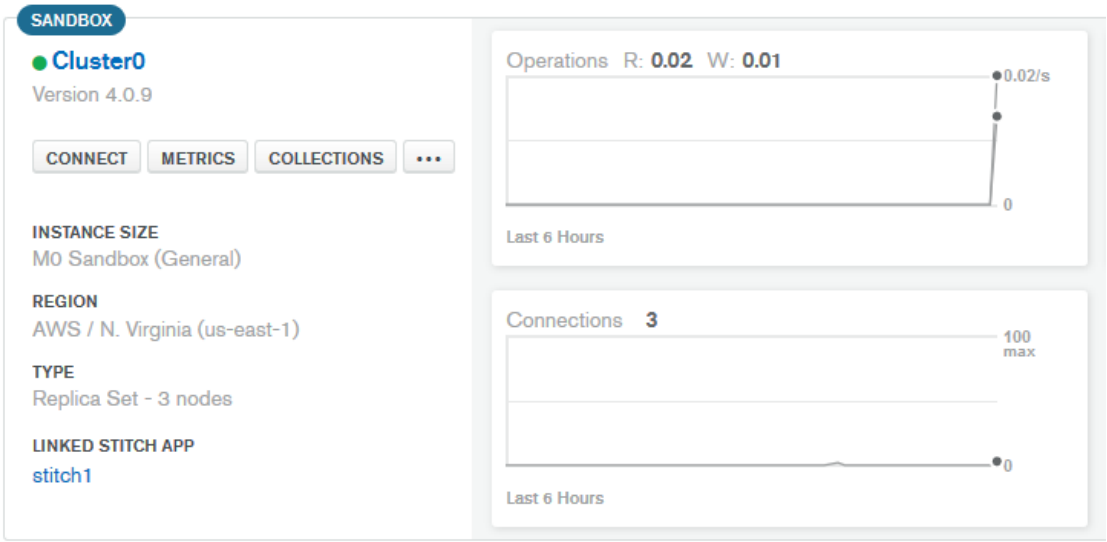


Fig 4.4: MongoDB Overview [9]

These are the relevant information about the data collection experiment of the environment. The storage environment is a cloud database names as MongoDB. Free version of MongoDB is used in this project. As it can be seen in Fig. 4.4, Mongo assigns the user a cluster and this cluster operates in a Sandbox. In the region section, it shows that Mongo uses Amazon Web Services and the current server is located in N. Virginia. The location is not directly used, but still it is a part of the environment. On the right side, there are two information windows about the number of connections and the average rate of read/write operations performed per second over the selected sample period. It is clear that 3 connections are done to the cluster (to the database) and the average rate of read/write operations were 0.02 and 0.01, respectively.

For the last part of this experimentation, one presentation screenshot is given in Fig. 4.5 from the user mode. Through this experimentation and environment design process, first the location is determined as ITU Ayazaga Campus. Second, a grid view is displayed on the campus. Then, measurements are done and more important collection points are chosen. Collected data is sent to the cloud database and finally in this step the data from the cloud database is fetched and the air quality is monitored on the map. This representation is for one precise date and time of the day. The colouring of the map may change according to the chosen date or time range.

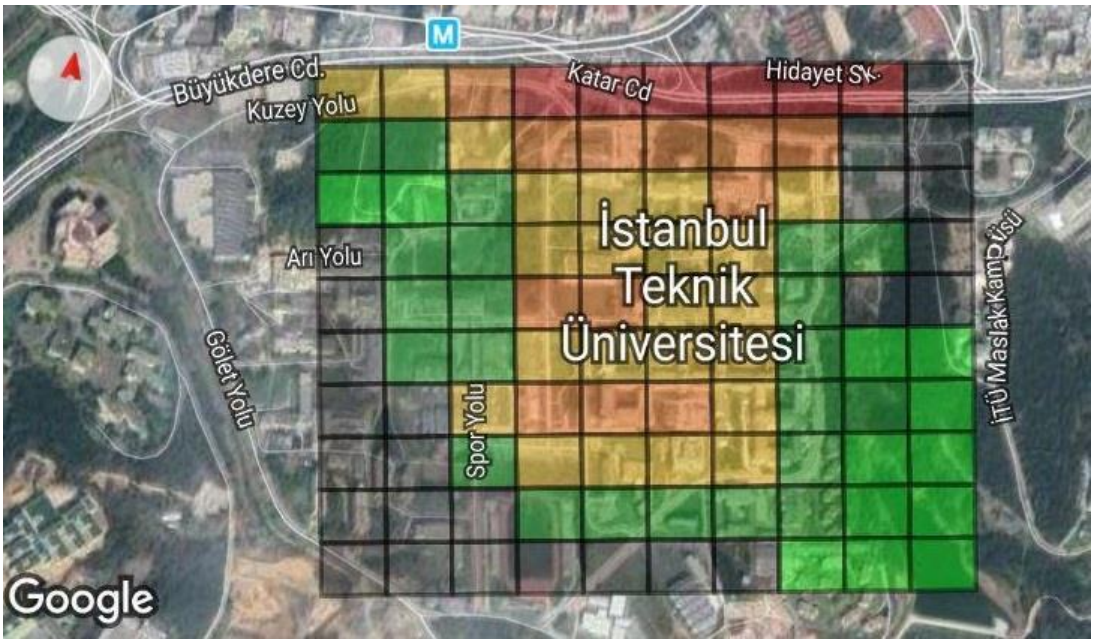


Fig 4.5: On map representation

5 Comparative Evaluation and Discussion

This project is not based on an algorithm realization or a machine learning task, so there are not any previous works in the literature which are equivalent to this project, for comparing in terms of quantitative measurements. Rather than this backward comparison, two types of evaluations can be done on this project. First one is checking whether the features of this finished project can satisfy the goals determined at the beginning. Second one is comparing the air quality results of different zones at different times.

At the start point of this project, some use cases were defined and some specifications were made in terms of performance of the environment. These use cases were about collector and user operations. Mainly, use case of collectors was successfully collecting from data the sensors and uploading them to the cloud database. On the other hand, related use case for the end users was freely walking around the application and monitoring air quality data for any date, any time. Both of these necessities are fulfilled.

In addition to use cases, some performance outcomes should also be evaluated. First step of this performance dynamics is the delay time while connecting to the Arduino by Bluetooth. When the Arduino is in the range of the collector mode, the connection can be established in under 5 seconds. Secondly, the traffic between these two devices are very small, so the needed bandwidth does not exceed the possible Bluetooth bandwidth. The environment is designed open-ended for future additions. It is easy to plug-in additional sensors or Arduinos to the system. The application support Android platform and it does not crash for any operation. Users can switch to any rational date and see the data on the screen in under 10 seconds. This means the cloud database is strong enough for satisfying the needs of this application, connection time, read and write times are in an acceptable range. There are 100 rectangles, 100 zones, on the campus map and in each update to the map, the relevant data from database is fetched for all of these rectangles so that they can change colours. MongoDB provides 50 GB of free space and every single air quality data covers around 0.4KB. The result of this comparison is that there is space for around 125 million entries, which is more than enough for the scope of the project. To sum up, all of these evaluations show that this work successfully satisfies the determined criteria.

The second and the last comparison is based on the air quality results of different zones. After long days of data collecting it was found that in addition to zones having different air qualities, some of the zones always have higher air quality averages than other ones. These zones are mostly near the roads. The worst results are taken from the upper border of the campus, because just after that border there is an always busy highway. Exhaust fumes of the cars and busses pollute the air and it reflects to the campus. On the other hand, the best air quality is around the “golet” (pond) area. There are no vehicles going around that corner of the campus and that point is not spoilt with buildings. Moreover, there are also differences among times of the days. Usually, in the mornings the air has the least pollutants and this pollutant level goes up through the night. This may be because of two things. First one is that emissions of air pollutants pile up from the morning till the night. Second one is plants make photosynthesis only in the daytime, after the night falls they stop photosynthesis and join carbon dioxide production.

6 Conclusion and Future Work

In this project, the effort was in developing a mobile application and creating an environment for collecting air quality data, transferring it, storing in this environment and presenting it on the application in various forms to the end user. First of all, a research was done for finding an appropriate air quality sensor. Secondly, Arduino UNO was used for integrating the sensor. For storing, MongoDB's non-relational cloud database was used. Finally, an Android application was developed for associating all of these components and creating an environment. After the system was fully ready, measurements were done along the campus and the results were monitored in the application to the user. The goal at the beginning of the project was achieved by the time this report was written.

There is a fully connected environment from the beginning point, the sensor, to the end point, the application. It is possible to measure, transfer, store and present the data in useful ways. However, this project has an open-ended context, some parts of the design can be improved as a future work with a higher budget, longer time and a larger team.

One of these possible improvements starts with increasing the number of air quality sensors and Arduino boards. When the numbers are increased, these double structures can be placed in various points of the campus and collection method can be switched to crowdsourcing. With this approach, the collector mode and the user mode of the application can be combined and every user inside the ITU Ayazaga campus can be both user and collector. Whenever the app is opened, the app can open the location and Bluetooth services of the phone, so that in the background the application can connect to the closest Arduino, receive the air quality data and send it to the cloud database. The need for authorized collectors, tiring and long lasting collection sequences vanishes when the project is improved in this way.

Secondly, the described system in the previous paragraph can be converted into a more autonomous environment. Again, the number of sensors and Arduinos are distributed to the campus. However, this time, Wi-Fi modules can be added to Arduinos. By this improvement, the Wi-Fi modules become the gateways and without the need of a connection to the application, the incoming data from the sensor can be uploaded to the cloud software. This time, the need for a collector is completely removed from the project. Moreover, the system gains the speciality of being always up-to-date. The data uploading time periods can be arranged by adding a few rows of code to the Arduino program. This period can be set to every hour or even to every minute. The number of data exploits and covers every spot of the campus. So, the map can be perfectly coloured in any time or for any chosen date.

Third future work is keeping the autonomy of the system, while decreasing the needed number of sensors and Arduinos. After long days of measuring, the critical spots in the campus can be determined. If the air quality of a particular zone in the campus is mostly stable for different days of the week, then there is no need for placing a lot of sensors into that zone, one or two could be enough. On the other hand, if the air quality of a particular zone is changing frequently, this means this zone contains critical spots inside, more sensors need to be placed for accurate measuring. So, by placing the sensors and Arduinos according to this critic/non-critic spot evaluation and using lower number of components, the system can still work properly.

As remarked before, this project is open-ended. Thus, for the last future work, the functionality of the application can be extended. Some relevant data to the air quality data can be added to

the system. This relevant data can be the temperature or the humidity of the atmosphere. The end user can see the air quality, temperature and humidity at the same screen. These values contain a correlation between each other, so the addition of this functionality can be effective. For this aim, new sensor or sensors should be integrated to the system. These sensors can either be standalone sensors with internet connections/Bluetooth modules like Texas Instruments' Sensortags or can be more basic analog output sensors like the MQ-135 sensor. After inclusion of this functionality, based on the correlations between air temperature, humidity and air pollution values and the historical data, air quality predictions for upcoming days can be done.

7 References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," in *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347-2376, Fourthquarter 2015.
[2] “What the Internet of Things (IoT) Needs to Become a Reality,” Freescale Semiconductor, Inc., ARM, Inc., 2014. [Online]. Available: https://www.nxp.com/docs/en/white-paper/INTOTHNGSWP.pdf
[3] M. Chen, J. Yang, L. Hu, M. S. Hossain and G. Muhammad, "Urban Healthcare Big Data System Based on Crowdsourced and Cloud-Based Air Quality Indicators," in IEEE Communications Magazine, vol. 56, no. 11, pp. 14-20, November 2018.
[4] R. Sharma. “Air Quality Monitoring.” Internet: https://www.hackster.io/ruchir1674/air-quality-monitoring-7c5bae, Apr. 22, 2018 [Feb. 14, 2018].
[5] Gironi, D. (2017, May 1). MQ gas sensor correlation function estimation by datasheet [Blog post]. Retrieved from http://davidegironi.blogspot.com/2017/05/mq-gas-sensor-correlation-function.html#.XOPBHo9S_IU
[6] H. Ali, J. K. Soe and Steven R. Weller (2015). A Real-Time Ambient Air Quality Monitoring Wireless Sensor Network for Schools in Smart Cities, presented at in IEEE First International Smart Cities Conference (ISC2), Guadalajara, Mexico, 2015.
[7] A Beginner's Guide to Air Quality Instant-Cast and Now-Cast. (2015, March 15). Retrieved from http://aqicn.org/faq/2015-03-15/air-quality-nowcast-a-beginners-guide/
[8] https://nevonexpress.com/MQ-135-Air-Quality-Sensor-Module-Pollution-Sensor.php, Date of access: 20 May 2019.
[9] https://cloud.mongodb.com/v2/5cb5fcb6014b76984e0d2bb1#clusters, Date of access: 22 May 2019.