A CROWD-SOURCING URBAN AIR QUALITY MONITORING SYSTEM WITH BIKES

By

Zhengxin Jiang

Kaiwen Hong

Haofan Lu

Haoqiang Zhu

Final Report for ECE 445, Senior Design, Spring 2021

TA: Xinyi Xu

May 2021

Project No. 01

Abstract

In this work, we designed a crowd-sourcing urban air quality monitoring system to be deployed on a public bike renting system, aiming at providing street air quality information for bike riders. To achieve this goal, we designed a hardware device with air monitoring sensors integrated for urban street-level air quality sensing. The device can be installed on a public bike via an off-the-shelf holder and connects to the internet through cellular network. A cloud system is designed that the data collected from the devices across the city is visualized on an air quality map and can be accessed by users via web browsers on their phones. Furthermore, an inference model has been designed to predict the air quality where there is no active device. We also designed a multi-source sustainable power supply scheme to cover the device power consumption.

Contents

1 Introduction

1.1 Objective

With the increasing emphasis on a healthy lifestyle, citizens are becoming more willing to ride bikes to nearby places. However, ambient air pollution comprises a significant health risk for bikers. Fixed official monitoring stations have been deployed to measure pollution concentrations at different scales. Unfortunately, stationary systems cannot meet the requirement of fine-grained monitoring and dynamic updating. To solve this problem, we identify two opportunities. First, urban air quality varies a lot spatiotemporally. Typically, air quality is relatively poor on congested roads because exhaust fumes are concentrated in this area. If bikers have air quality information on specific streets, they could choose the routes with better air quality to go. Second, the public bike system has been strongly-invested by local governments at all levels in China in the past few years. Large-scale convenient bike renting systems have been well-established by now, which is promising in serving as the air-monitoring device holder for mobile sensor networks. With these two key observations, we plan to design a crowd-sourcing urban air quality monitoring system, deploying on the public bike system, which aims at providing detailed street-level urban air quality information for bikers.

1.2 Background

Our project has both commercial value and social significance. Public bike system is growing more and more popular in recent years. According to a report from China Industry Information website, by July of 2015, the number of public bike systems has exceeded 52, with a total market size of over 1.2 billion yuan. Thriving bike-sharing companies, such as Meituan and Hello Bike, also provide great commercial markets for our monitoring system in recent years .

On the other hand, for biker users, air contaminants, such as particulate matter (PM), are widely concerned for the severe diseases resulting from them. The most common effects caused by exposure to particulate matter include irritation to the respiratory tract, lung diseases. Our system can inform bike users of heavily polluted areas, to help them avoid potential health risk.

Crowd-sourcing is becoming a new paradigm with the rapid development of Internet-of-Things (IoT). It is easier than ever to collect data from ubiquitous sensing devices. Data mining techniques and algorithms can be applied to the data collected to draw critical conclusions. Many past works have proposed to collect air quality data from stationary and mobile sources to get wider sampling coverage and finer sampling granularity. For example, OpenSense [1, 2] proposed to deploy sensors on mobile vehicles and stationary monitoring stations to obtain the urban air quality map. Similarly, BlueAer [3], AirCloud [4], Gotcha [5] also propose to leverage the mobility of buses and taxis to get air quality maps. Our advantage lies on that we use the public bike system, which has a large and stable user group in China and all bikes are managed by official organizations. Therefore, the deployment of air quality sensing devices on public bikes can be done in a collective manner.

1.3 Design Overview & Visual Aid

Figure 1 illustrates the way our system works. Bike users ride bikes on urban roads. Our devices, which are installed on the handle of bikes, connect to users' mobile phones through hotspot access points. Air quality and GPS data is uploaded to the remote server through cellular network. The remote server generates the air quality map according to the air quality data inference algorithms and then updates the air quality map on the website. Users can find this map on their mobile phones to see the street-level air quality.

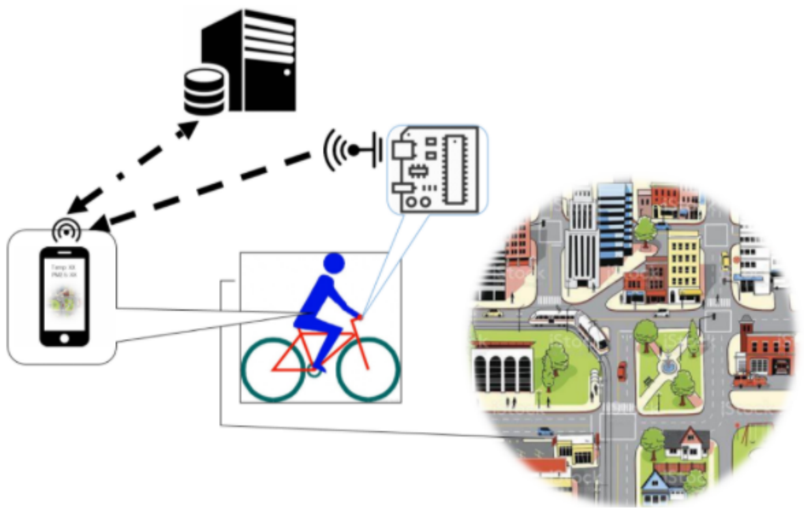


Figure 1: System Design Overview

1.4 High-level Requirements

- The on-bike air quality monitoring device should have stable connection with the database and be able to collect and upload sensor, GPS and battery voltage data. The device should measure air contaminants (particulate matter, CO and NO2) concentration correctly.
- The power module should supply stable 5V output. The power source should be able to cover the device's power consumption.
- The air quality map web page should be able to achieve data from server database and display a real-time street-level air quality map.

1.5 Block Diagram

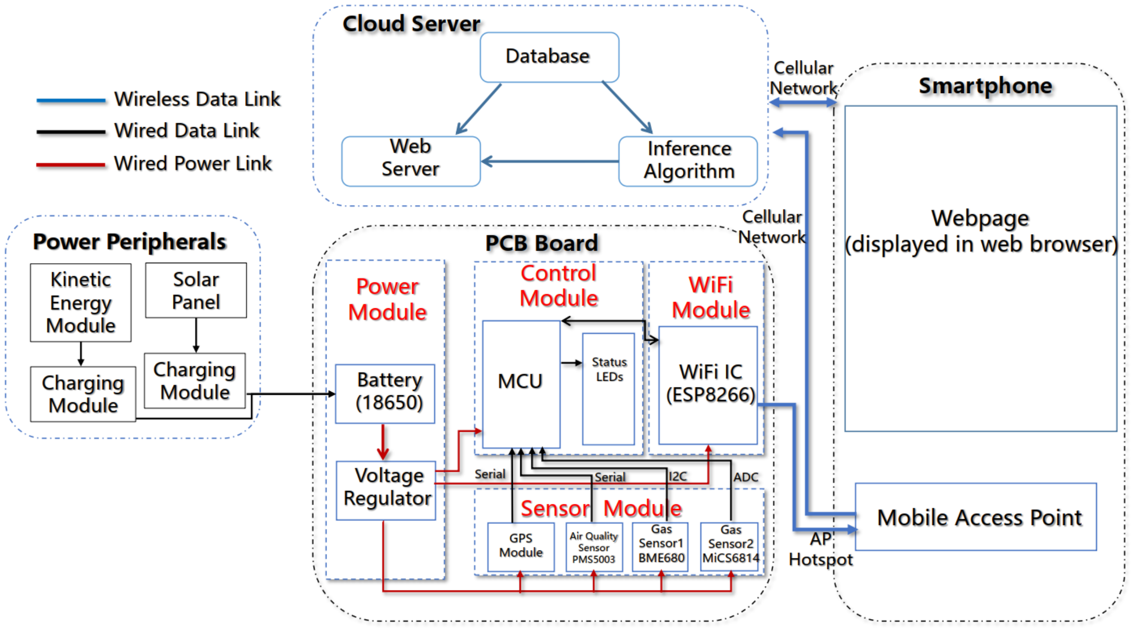


Figure 2: Block diagram of our project

Our system block diagram can be divided into three main components: the hardware device, user's mobile phone and cloud server. The hardware device is a PCB board with sensors and micro-controller. The sensor data collected will be uploaded to the cloud server via hotspot on user's mobile phone. Cloud server will process the data and formulate into an map. This map is embedded in a webpage and can be accessed via any browser on user's mobile phone or laptop. Figure 3 further illustrates the data transmission in our system. The power module for the hardware device is particularly designed to inte]grate the kenetic energy charging and solar energy charging functionalities.

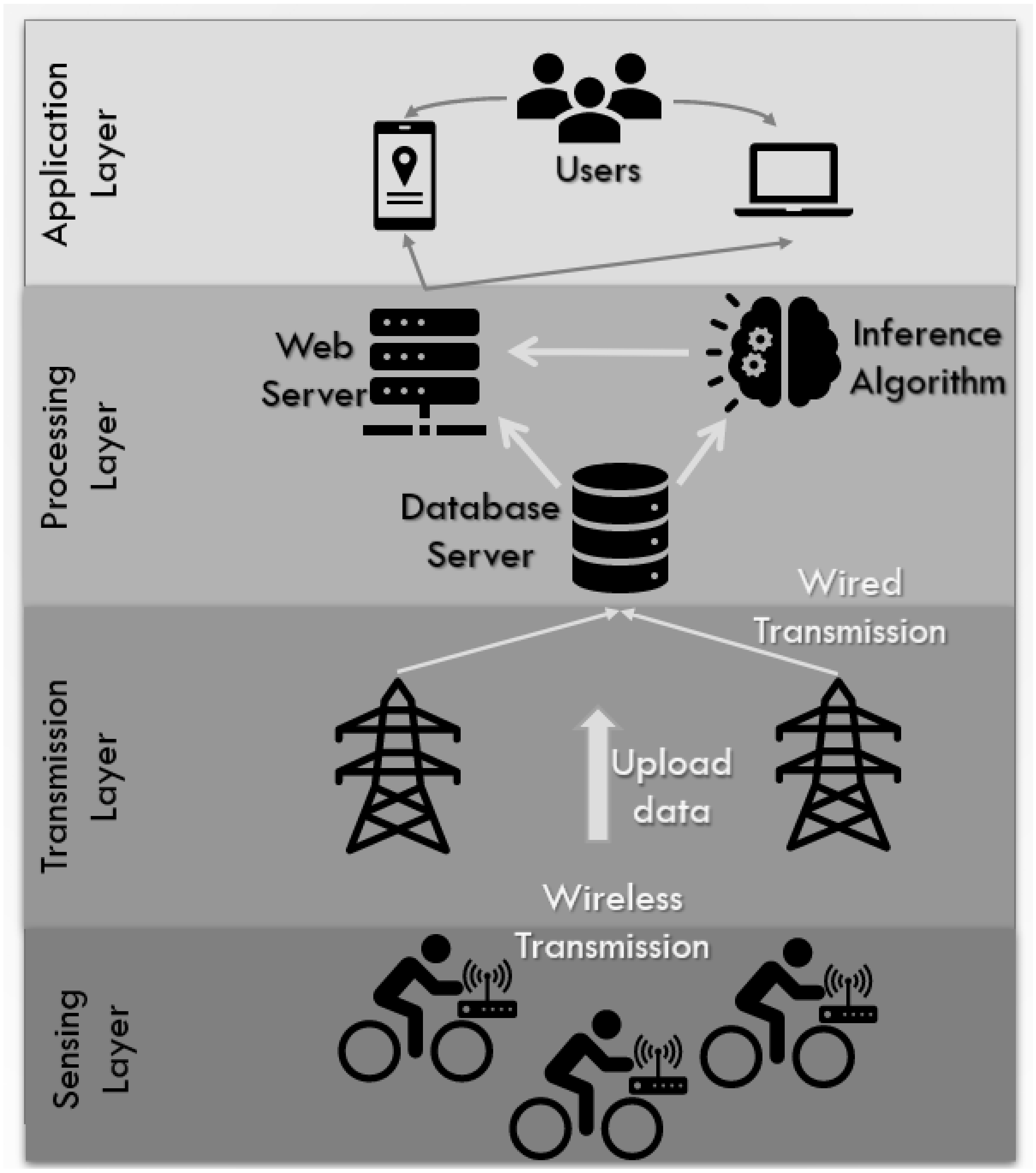


Figure 3: Four layers of the entire system

2 Web Design

We designed a webpage as a User Interface (UI) to display the air quality map. Basically, following requirements should be satisfied:

- A cloud server should be set up to store and process data. Data collected from hardware devices should be transmitted and stored in the database for further processing. The inference algorithm also runs on the server to generate maps.
- The webpage displays a street map for the city, which the user is localized. Users can manipulate the map (zoom in/out and rotate) for his/her preference.
- A overlay layer should be added to the map to indicate air quality on each street.
- Various air quality indexes (PM1, PM2.5, PM10, CO, and Temperature) should be displayed on the map.
- Air quality on each street at several future moments predicted by inference algorithm should be displayed for user's reference.
- The air quality statistics measured by the user's device can be displayed on the webpage in real time.

Guided by the above design requirements, the software design work can be divided into two parts: backend and frontend.

2.1 Backend Design

The backend part consists of the database management, web server configuration and inference algorithm implementation. The inference algorithm implementation will be discussed in detail in section 4. In this section, we will focus on discussing the other two parts and the interfaces among them.

2.1.1 Database

We adopt *InfluxDB v1.8* as the database management software. For the two reasons:

1. InfluxDB is specialised in operating time series data. Every input data point is automatically associated with a Unix time point. Our input data streams from hardware devices strictly follow the time sequence, which is well-fitted to InfluxDB's advantage.
2. InfluxDB has well-developed interfaces with many mainstream programming languages. The developing languages: Python and PHP are well-supported.

Within the database, each index we measure has an ad hoc category to store. The retention policy of the database is set to one week, as the model is trained once a week, only with the data acquired within one week.

2.1.2 Web Server

For web server, we choose Nigix to hold the website. Nigix is a widely adopted web server software that handles the delivery of webpage files. A URL is associated with the web server: www.ece445-air-quality.top. Users can find our website via the URL in their browser.

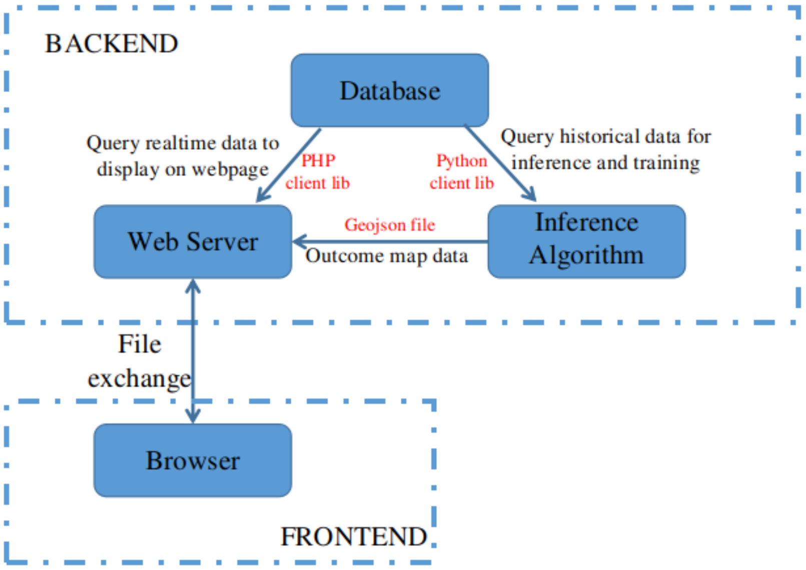


Figure 4: Software block diagram

2.2 Interfaces

The interfaces between backend components are designed as shown in figure 4. InfluxDB provides official client libraries for Python and PHP. We leverage these libraries to build interfaces with database. The inference algorithm is written in Python; therefore, we use the Python client library to directly query data from the database. The PHP script running on the web server needs to display the real-time air quality information. We use the PHP client library to realize this function.

The interface between the inference algorithm and web server worth more discussion. The previous work [6] has a map-generating function in the inference algorithm, which outputs image files to be overlaid on the street map. However, this approach is not effective for our application for the following reasons:

1. The image files cover the entire area, where not only streets but also surrounding buildings are covered. Our project focuses on the air quality in the street areas, and the surrounding buildings are not in our scope.
2. The redundant information carried by the image files takes more bandwidth to transmit; therefore, reduces the processing rate. As our map is going to cover the entire city, a large image files should be avoid to ensure the quick loading of webpage.

Therefore, we choose to use GeoJSON [7] as a file output format from the inference algorithm. GeoJSON file only contains the information needed to render the map and leave the map generation task to frontend, which effectively reduces the burden of file transmission and increases the process rate. The tradeoff of taking this approach is that the work of rendering map is left to the frontend, which means the user's device (mobile phone and laptop) must take more computational resources to process the GeoJSON file and render the overlying layers. However, considering the increasing performance of mobile end devices, this drawback should be tolerable. Our experiments on HUAWEI P40, Apple iPhone 12, and Samsung Galaxy S10 all demonstrate fluent rendering of the map on the frontend.

2.3 Frontend

On the frontend, user's browser can fetch the webpage file from backend once the URL is requested. The work on the frontend mainly focuses on the layout design and the interaction with backend server.

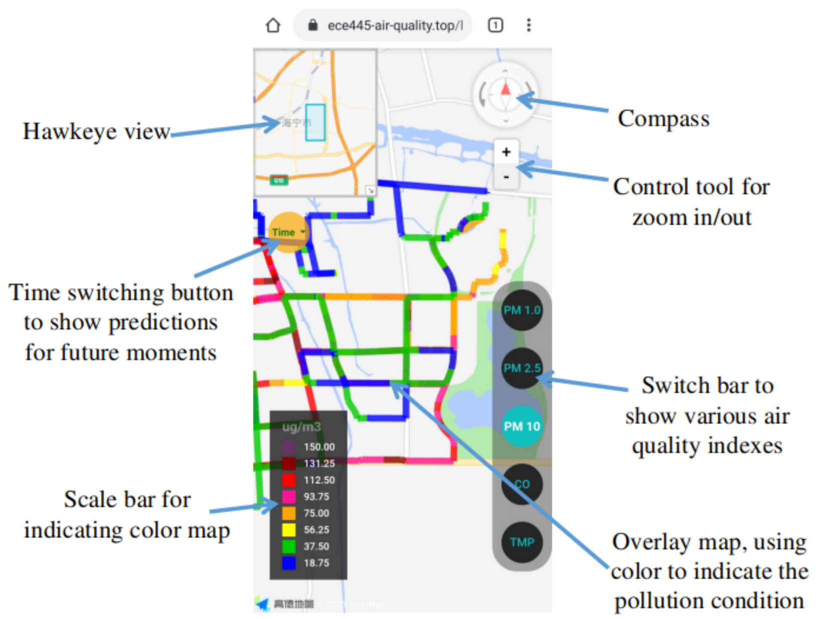

ece445-air-quality.top/
Hawkeye view
Compass
Control tool for zoom in/out
Time
Time switching button to show predictions for future moments
PM 1.0
PM 2.5
PM 10
CO
TMP
Switch bar to show various air quality indexes
ug/m3
150.00
131.25
112.50
93.75
75.00
56.25
37.50
18.75
Scale bar for indicating color map
Overlay map, using color to indicate the pollution condition
高德地图

Figure 5: Webpage layout

2.3.1 Layout Design

For frontend layout design, we use Amap JS API [8] and Loca JS API [9]. The designed webpage for mobile end is shown in figure 5. There is a hawkeye map at the upper left corner to show a broader view. On the upper right corner is the compass and control top for zoom in and out. User can also control the map via figure operations on the screen. The switch bar at the bottom right corner is for displaying map of various indexes. User can touch different buttons to view the map of any of the indexes we measured. The scale bar at the bottom left corner is intended to give quantitative illustrations for the color we use on the map. Users can get an idea about the colors and corresponding pollution conditions. There is a time switching button below the hawkeye map, users can select to view the predicted air quality at several future moments. The air quality is visualized on the map using colored poly-lines.

3 Data Processing

3.1 Design procedure

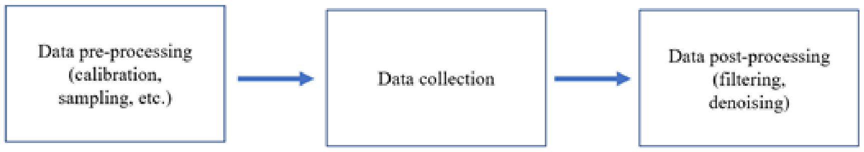

Data pre-processing (calibration, sampling, etc.)
Data collection
Data post-processing (filtering, denoising)

Figure 6: Workflow of data collection and processing

The first step is air quality sensor calibration. We need to make sure that all sensors are calibrated by using a professional air-quality measurement device called air-quality egg provided by our advisor. To calibrate the sensors, we carried our device and the air-quality egg to different places including construction site, traffic ways and campus. For temperature measurement, the air-quality egg can update the data every 1 minute; for PM measurement, the air-quality egg can update the data every 2 minutes and for CO measurement, the data is updated every 5 minutes but for NO_x measurement, the data is sparse (sometimes once in 20 minutes). We used linear regression to get the relationship between the measurement data and the calibrated data for simplicity.

3.2 Calibration

So far, we have deployed two devices on the bike and finished the calibration procedure. As indicated in the last section, we use linear regression to get the relationship between the measurement data and the calibrated data. We used least-square method for calculating the best-fitting line for the measured data [10]. Assume we have this first order linear model:

$$y = b_0 + b_1 x + \epsilon \tag{1}$$

where y is the dependent variable, x is the independent variable, b_0 is the Y-intercept, b_1 is the slope of the line and ϵ is the error variable. The least-square method is to minimize:

$$\sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{N} \epsilon^2 \tag{2}$$

The resulting equation for b_0 and b_1 is:

$$b_1 = \frac{n \sum_i x_i y_i - \sum_i x_i \sum_i y_i}{n \sum_i x^2{}_i - \sum_i x^2{}_i} \tag{3}$$

$$b_0 = \frac{\sum_i y_i - b_1 \sum_i x_i}{n} \tag{4}$$

For sensor calibration, we went several places including construction sites, traffic ways and campus. The resulted regression line of device 1 is shown below for each air-quality data. From the graph we can get the relationship between the measurement data from our device and the relatively accurate data from a professional device using linear regression. After the calibration, we can get accurate and consistent measurements for air quality data.

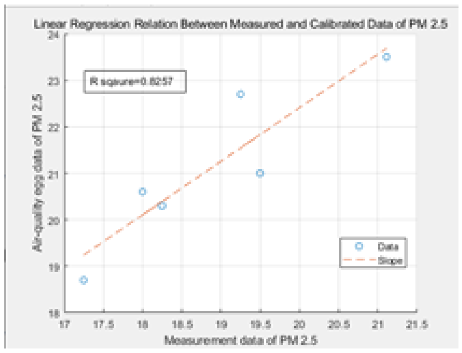


Figure 7: Calibration Relationship of PM 2.5

3.3 Data Collection

Currently, we have deployed two devices on the bike and rode across streets around the campus. The sampling rate of our device is 5 seconds. We have collected the environmental data for three days and one hour per day, namely in total three hours. The bike moving trajectory is recorded in “gpx” format, which includes the GPS information, time slot and speed. The total trajectory is 60 km long. We analyzed the collected data by segmenting the trajectory into pieces of 500 m long. We used the open-source data from AMap [8], which is a free, crowd-sourced, and user-generated online mapping system. The corresponding Software Development Kit (SDK) is used for development.

4 Inference Algorithm Design

4.1 Introduction and Motivation

The static air-quality monitoring stations are commonly deployed in urban areas to get the air quality data, yet the size and cost of the large station were so high that they cannot be deployed sufficiently enough for the fine-grained air pollution map [5]. Thus, mobile sensing networks have been proposed frequently in recent years, including OpenSense [1, 2], BlueAer [3], AirCloud [4] and Gotcha [5]. However, although the mobile sensing networks could provide larger air-quality data with wider range and granularity, the collected samples still cannot cover the whole spatial-temporal area since the collected data are scattered. Therefore, an inference algorithm is needed for recovering the fine-grained air pollution condition and further helping researchers to investigate and comprehend the environmental condition. [11].
The workflow graph below gives an overview of what our project does from the algorithm perspective. The workflow can be divided into offline learning, online interference, and simulation for determining key parameters of experiments. The workflow starts from feeding open-source data into the model, train and evaluate it and then select the best algorithm. Then we build a simulation to determine how much data we need to generate an air-quality map for a certain area. If the resulted number is beyond our ability, we will narrow the range of the air-quality map for this process. After that, we need to collect data in Haining city and do the processing step as mentioned in previous section. When these procedures are done, we can feed the air-quality data into a selected model to train it. The weights trained on the open-source dataset can be used for the weight initialization of the inference model of Haining city. The outputs will be used for the air-quality map visualization.

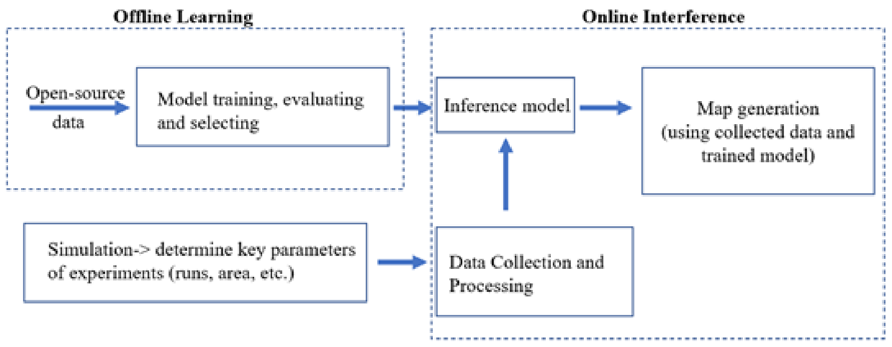

Offline Learning
Online Interference
Open-source data
Model training, evaluating and selecting
Inference model
Map generation (using collected data and trained model)
Simulation-> determine key parameters of experiments (runs, area, etc.)
Data Collection and Processing

Figure 8: Workflow of Inference Algorithm

4.2 Related Works

There are two types of inference algorithm for urban air-quality monitoring, the first type is based on static monitoring stations and the second is based on mobile sensing systems. The inference algorithm of the first type can be divided into two groups: deterministic model-based algorithms and data-driven algorithms [11]. Deterministic model-based algorithm is based on the assumption that physical dispersion process, chemical process can model the real-world air pollution scenario [12, 13, 14]. However, they require large computational resources and complex prior knowledge of meteorology and pollution sources [11], which we do not have. Data-driven algorithms are used to infer the unobserved area using spatial interpolation methods for example K-nearest-neighbor interpolation [13] and land-use regression (LUR) [13]. However, the methods are not perfectly convincing since the auxiliary information might change over time and the mathematical relationship between factors and air-quality data is not reliable [11]. The inference algorithm of the second type includes interpolation method [15], probability concentration estimation method (PCEM) [16], Gaussian Process Regression (GPR) [17], artificial neural networks (ANN) [18], autoencoder framework [2020Fine] and reinforcement learning method (RL) [19]. The description of the methods is summarized in Table 1. We further summarize the advantages and disadvantages of the above methods in Table 2.

Module Name	Description
Interpolation	Uses linear/gaussian interpolation to infer the unobserved air-quality data using nearby data.
PCEM	Models the particle motion based on the concept of "random walk".
GPR	Models the pollution field as multivariable Gaussian distribution and considers features of GPS coordinates, location-related humidity, temperature, and Point of Interests (POI).
ANN	Uses artificial neural networks which takes both spatial and temporal data as input.
Autoencoder	Uses deep autoencoder model with convolutional long short-term memory (ConvLSTM) structure.

Table 1: Description of Inference Algorithms

Module Name	Advantages	Disadvantages
Interpolation	Easy to implement	Accuracy highly depends on nearby location; requires regular and dense sampling and low sensing error
PCEM	Can suit for variable weather conditions.	The probability requires all the samples are observed at the same time, but it is unrealistic.
GPR	Non-parametric method which can model arbitrary functions	The computational complexity is $O(N^3)$, which means with data increasing, it requires huge computational resources.
ANN	Can solve the data sparsity problem.	The neural network is more like a black box which lacks rigorous theoretical proof of why it works.
Autoencoder	Can suit for variable weather conditions; better captures the spatiotemporal dependencies with nonuniform sample distribution.	The neural network (ConvLSTM) is more like a black box which lacks rigorous theoretical proof of why it works; the autoencoder is relatively hard to implement.

Table 2: Advantages and disadvantages of inference algorithms

4.3 Algorithm Overview

4.3.1 Preliminary

In this section, we will introduce some basic concepts used in the inference algorithm including meteorological data, point of interests(POI), road network and grid.

Definition 1: Meteorological data: Meteorological data refers to the weather condition of a city in a certain period of time including temperature, humidity, pressure and weather. The weather condition includes snowy, cloudy, rainy, foggy and haze while in this project we only consider the condition of rain.

Definition 2: Point of Interests: Point of interests refers to a specific venue that have some functionalities (e.g., a school).

Definition 3: Road Network: Road network refers to a combination of road segments, each having its individual identifier $r.id$, length $r.len$ and level $r.level$.

Definition 4: Grid: Grid refers to the smallest unit we want to infer and we assume the air quality inside the grid is uniform. Each grid g has its geographical location $g.loc$ and air quality measurement.

4.3.2 Problem Formulation

In this section, we will introduce the goal of the inference algorithms. Suppose we have observed the air quality data recorded in a spatiotemporal format, we establish a matrix $\boldsymbol{A} \in \mathrm{R}^{M \times T}$, where the element $\boldsymbol{A}_{i,j}$ refers to the air quality data collected in location g_i and timestamp t_i. Note that this matrix is extremely incomplete, as in *Table to be done: complete of matrix in the dataset* indicates. The unobserved entries are padded with zero. Our goal is to estimate the unknown entries in the matrix. In other words, we need to complete the pollution matrix and get the complete air quality map given we have limited observations.

4.3.3 Method

This is directly copied without paraphrasing! The k-Nearest Neighbor (kNN) algorithm is a useful machine learning technique for classification and clustering. To get the baseline result, we adopt the classic kNN algorithm in the screening process, where both spatial and temporal kNN are included[20]. The temporal kNN selects k locations with known values, based on spatial Euclidean distance. The temporal kNN selects k locations with known values, on the condition that these k locations have the most similar temporal change patterns. We use the historical data to serve as the input to train the Deep Neural Networks (DNN) to predict the unobserved value. With the help of kNN algorithm, the complexity of DNN will be decreased significantly.

4.3.4 Framework

In this project, we decide to use a Variational graph auto-encoder (VGAE) framework proposed in [21], which represents the latest progress in the field of air quality inference. Consider n nodes of the air quality graph constructs a continuous random vector $\mathrm{x}_n \in \mathrm{R}^T$, we then set a latent variable $\mathrm{z}_n \in \mathrm{R}^m$. That is, the $\mathbf{X}$ is assumed to follow the conditional distribution of $\mathbf{X}$ given $\mathbf{Z}$. Then we consider approximating $q(\boldsymbol{Z}|\boldsymbol{X})$, Z can be interpreted as a latent representation of the observed X, which is thought be a probabilistic encoder and the generative process characterized by $p(\boldsymbol{X}|\boldsymbol{Z})$ is thought be a probabilistic decoder.

5 Mechanical Design

Since our device needs to be placed on a bike, mechanical designs of device fixing and packaging need to be done. In this section, we will introduce our two mechanical designs: the device holder and the device container. The device holder is fixed on the handler of the bike, where the device is attached. The device container is a box covering the device for rain protection.

5.1 Device Holder

The device holder is designated for the part attaches our device to the bike. The requirement is that it can resist vibrations when the bike is passing some bumping areas, for example the deceleration strips. In that case, we did not consider using 3D printing since it is not robust enough for daily use. Instead, we adopted a mature commercial product which originally aimed at attaching mobile phones on a bike. Figure 22 is the Computer-Aided Design (CAD) we drew for that device holder using Solidworks 2018. The advantage of this design is that the device can be tightly fixed in the device holder by adjusting the length of the middle bolts.

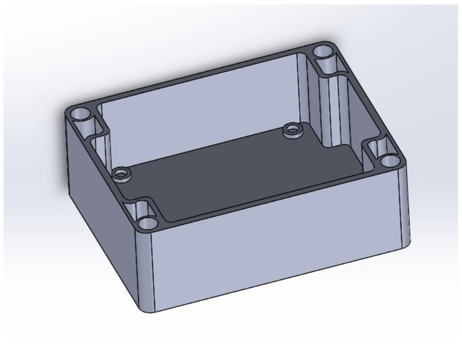


Figure 9: CAD of container

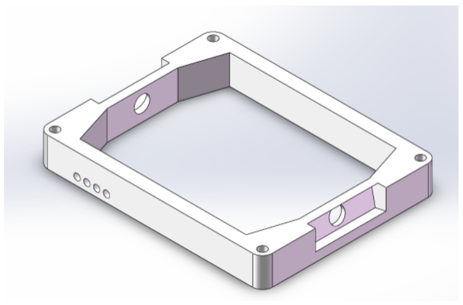


Figure 10: CAD of 3D printing parts

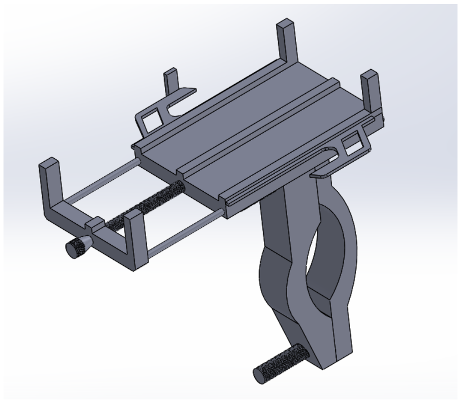


Figure 11: CAD of device holder

5.2 Device Container

The device container is designated for protecting the sensors, batteries and other electrical components from rain and other damage. The requirement of the container is that it should allow air to flow through the inner space while prevent raindrops. Also, since we want the container to be robust enough, we adopted a commercial product plus 3D printing plan. The majority of the container is based on a commercial container as Figure 9 shows. However, it is almost sealed up so we designed a structure in between, as Figure 10 shows. There are 2 special holes on the front and back of the 3D printing parts but with a slope to prevent rain from damaging the electrical parts inside the container. There are also 4 holes on side for the LED installation.

6 Hardware Device Design

6.1 Hardware Circuit

The hardware part is a compact device that collects and upload air quality and GPS data to the cloud server. The main components of the hardware circuit include a stm32f103 microcontroller, air quality sensors (PMS5003, MICS6814 and BME680), an ESP8266 Wi-Fi module, a GPS module, a power supply subsystem which consists of a multi-source battery charge-discharge chip and a 5V regulator, and a battery voltage monitoring circuit.

6.1.1 Sensor Modules

PMS5003
PMS5003 is a digital particulate matter (PM) concentration sensor produced by Plantower. In our project, the PMS5003 sensor measures the PM1, PM2.5 and PM10 concentration in atmospheric environment. The PMS5003 sensor communicate with the stm32 microcontroller by serial interface.

MICS6814
MICS6814 is a triple-channel gas sensor which can detect CO, NO2 and NH3. The outputs of MICS6814 gas sensor are analog signals, which can be read by STM32F103's built-in analog-to-digital converter (ADC). Three analog pins on stm32 can read CO, NO2 and NH3 measurements individually (pull-up resistors are needed for each analog pin to read correct input voltage).

BME680
BME680 is a high-accuracy temperature, humidity, pressure and volatile organic compounds (VOC) sensor produced by Bosch. The BME680 sensor supports I2C and SPI protocol for data transmission. In our project the sensor communicates with the stm32 microcontroller using I2C protocol.

6.1.2 Wi-Fi Module

The ESP8266 Wi-Fi module is used to support the air quality data upload function. The hardware device relies on cellular network to connect our cloud server, and the ESP8266 Wi-Fi module will connect the hardware device to user's smartphone for accessing cellular network.

6.1.3 GPS Module

The NEO-6M GPS module is used to record the real-time device location.

6.2 STM32F103 Running Code

The stm32f103 microcontroller is coded to run the routine of data collecting and uploading correctly. The software code is developed in Arduino IDE. Arduino IDE has mature support on stm32 series microcontroller, the software code can be directly compiled and downloaded to the stm32 microcontroller with correct configuration in Arduino IDE. Compared with another develop method using stm32cubeMX, Arduino IDE compiles much smaller program file, and this is significant for stm32f103 series with only 64KB flash. Therefore, consider the code readability, the development difficulty and the memory space utilization, we choose to develop the microcontroller software in Arduino IDE.

6.2.1 Module Interfacing Details

The stm32f103 microcontroller interfaces with other modules to collect and upload data in every loop. This section shows all interfacing details between the stm32f103 microcontroller and individual modules. Figure below shows the pinout diagram of the microcontroller.

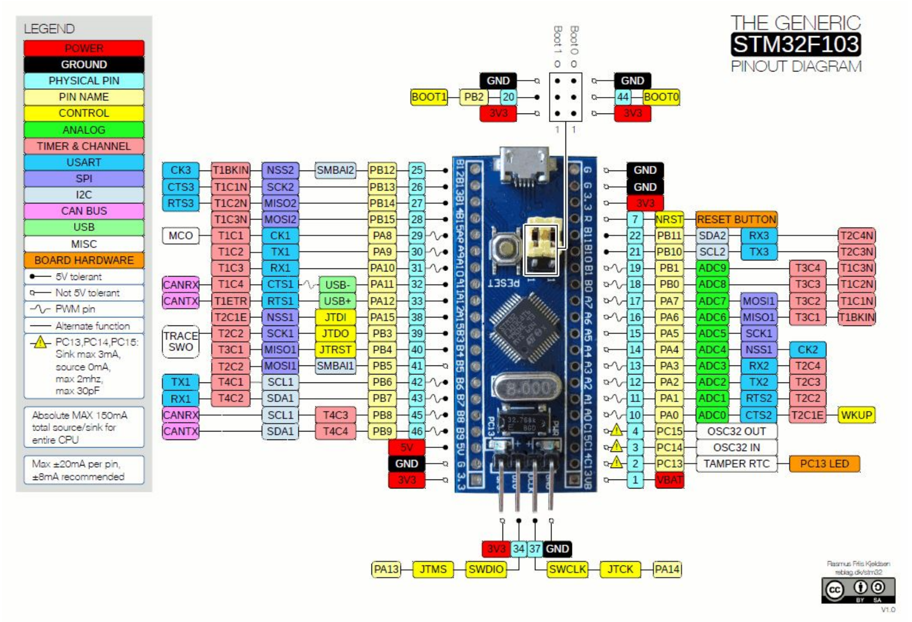


Figure 12: STM32F103 pinout diagram [22]

PMS5003

The PMS5003 sensor is connected to the stm32f103's third serial port (TX3-PB10/RX3-PB11). For every measurement, the PMS5003 sensor will send a data frame through the serial port. The start of the data frame is defined by two special bytes 0x42 and 0x4d, and the total length of the data frame is fixed as 32 bytes. On the side of the microcontroller, the special bytes 0x42 and 0x4d should be first found in serial port, then the microcontroller will continue to read until 32 bytes and compute the checksum to validate the data frame. Once get a valid data frame, the PMS5003 reading is done.

MICS6814

The MICS6814 sensor is connected to stm32f103's analog pins. Three measurements (CO, NO2 and NH3) are individually read by three analog pins (PA4, PA5 and PA6).

BME680

The BME680 sensor is connecting stm32f103's I2C port (SCL1-PB6/SDA1-PB7). The Adafruit BME680 library is used to initialize the I2C device and read BME680 measurement values.

ESP8266

The ESP8266 Wi-Fi module is connected to stm32f103's second serial port (TX2-PA2/RX2-PA3). The stm32f103 microcontroller controls the Wi-Fi module by sending AT commands through the serial port.

NEO6M GPS

The NEO6M GPS module is connect to stm32f103's first serial port (TX1-PA9/RX1-PA10). The GPS module will feed the serial port with NMEA format GPS data. The Tinygps++ library is used to extract latitude and longitude information from the raw data frame.

6.2.2 Test Mode

The microcontroller software is designed to allow the user to turn on test mode to see output prints on computer screen. Since stm32f103 has totally three serial ports which are occupied by PMS5003, Wi-Fi and GPS module, we choose to make the GPS module share the same serial port with the message print function in test mode. That means when the test mode is turned on, the GPS module will be temporary disabled to allow the serial print function. The choice makes sense because GPS will have no signal during indoor tests. GPS module can be individually tested if needed.

6.3 PCB Design

To meet with the size requirement for a mobile device installed on bikes, we design to put components on two individual PCBs and connect the two PCBs using pin header. In this way the dimension of the PCB is controlled in 86mm*48mm.

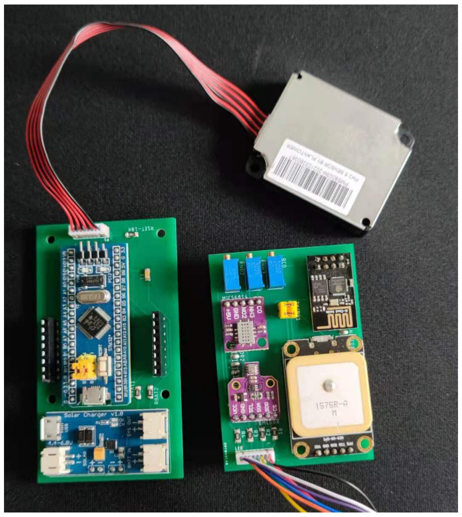


Figure 13: PCB boards before assembled

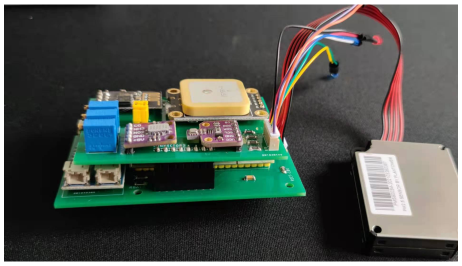


Figure 14: PCB boards after assembled

7 Power

We design a multi-source power module for our devices consider that our devices are long-term installed on bikes and it is inconvenient to manually change the batteries. The power module consists of three main components:

- 18650 Li-ion battery
- A solar charger
- A kinetic charger

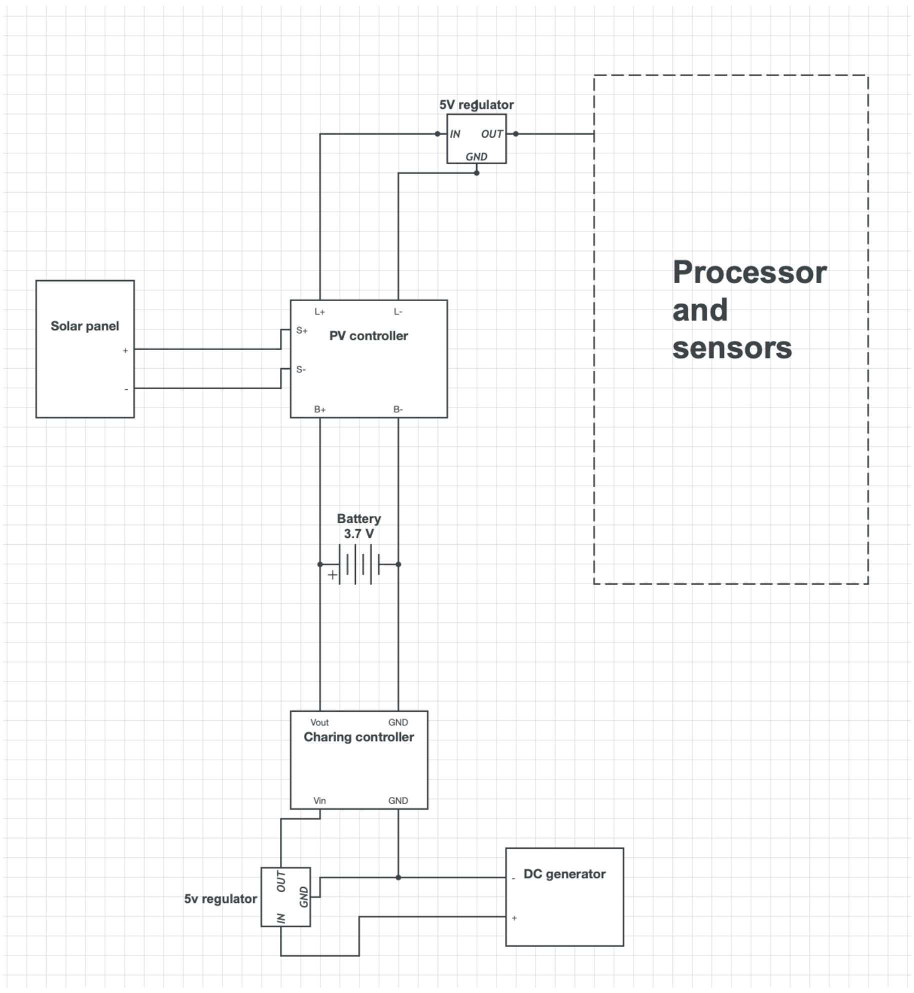


Figure 15: Power module block diagram

The size of the solar panel is 153mm x 99mm and it can provide 2.5 watts output at the optimal weather condition. The PV controller is used to control the charging process and it use the CN3065 linear charger chip as the control unit. For the kinetic charger, it can provide power when bikes are running. In theory, it can provide a power range from 0 to 5 watts. The kinetic energy module contains a small motor and a regulator for standard 5V output.

7.1 PV Controller

Below is the diagram of the PV controller. The input power should be in the range of 4.4V to 6V. It has two LED to indicate the working status. If the Red LED is on, it means the battery is being charged. If the Green LED is on, it means the charging process is done. The highest charging current is limited by the CN3065 chip, which is 500mA. The ISET pin is used to control the output voltage. To make the output voltage to be 4.2V, the resistor RISET should be 2K ohm.

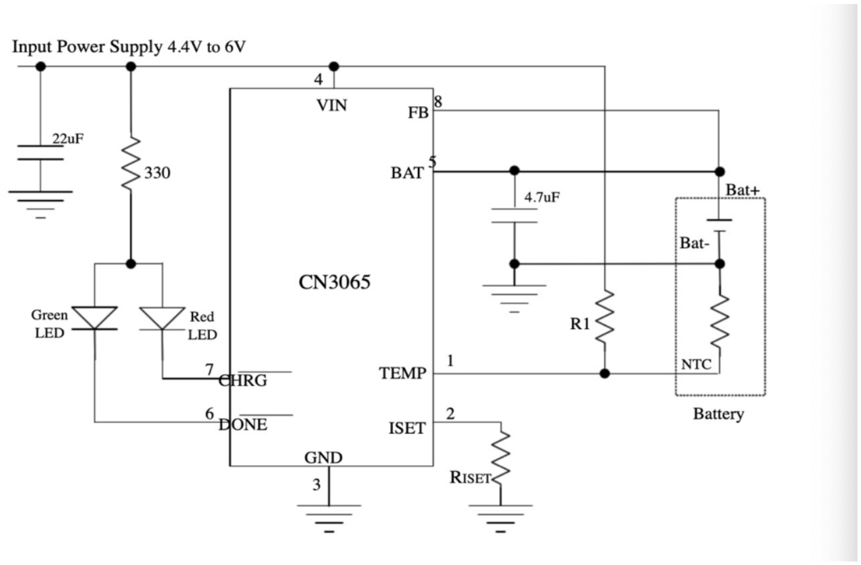


Figure 16: PV controller circuit

Figure 17 shows how CN3065 control the charging current and charging voltage in different phase.

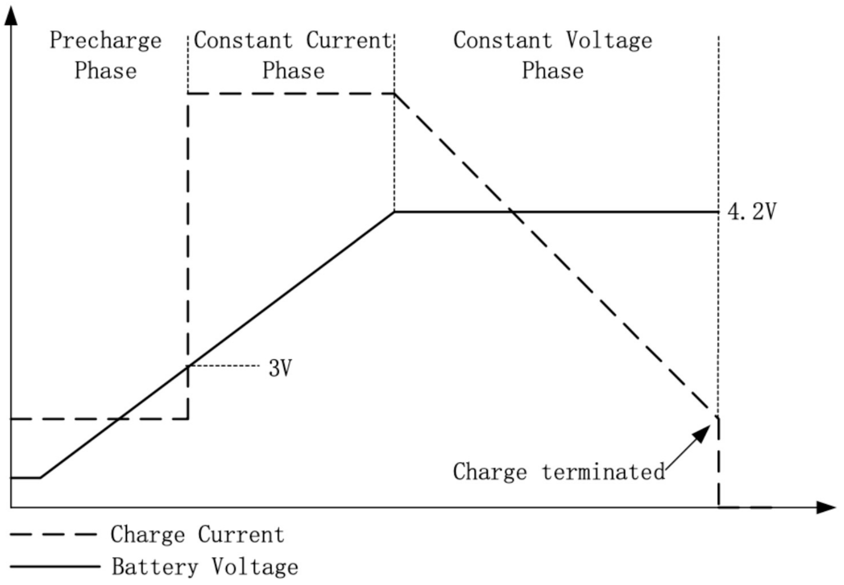


Figure 17: CN3605 charging phase

7.2 Kinetic Energy Converter

We add a kinetic energy converter to our power module in case that solar power is not stable and the power it can provide is determined by weather condition. Figure 18 shows the installation of the converter. When the bike is being riding, the wheel can drive the motor and the motor transform the kinetic energy to electric energy. As shown in Figure 15, there is a 5V regulator for standard 5V output. The output current depends on the riding speed. 500mA current can be produced for 18km/h speed, which is enough to cover the device power consumption.

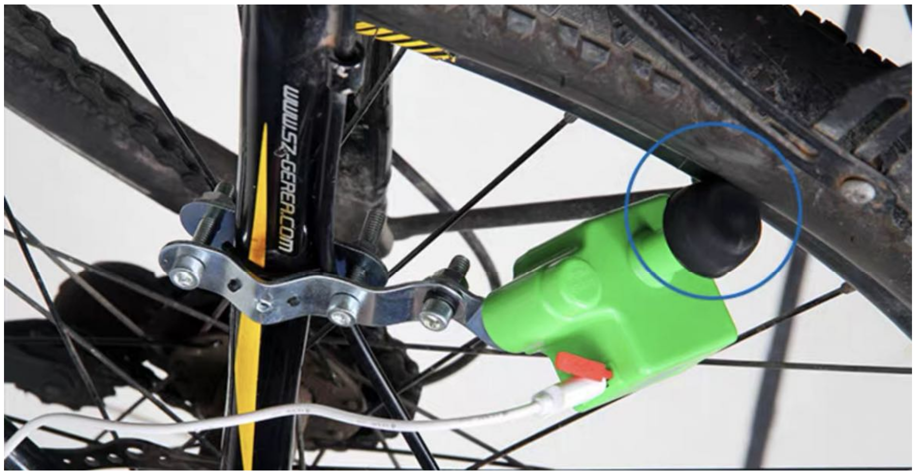


Figure 18: Installation of kinetic energy converter

7.3 Solar Panel

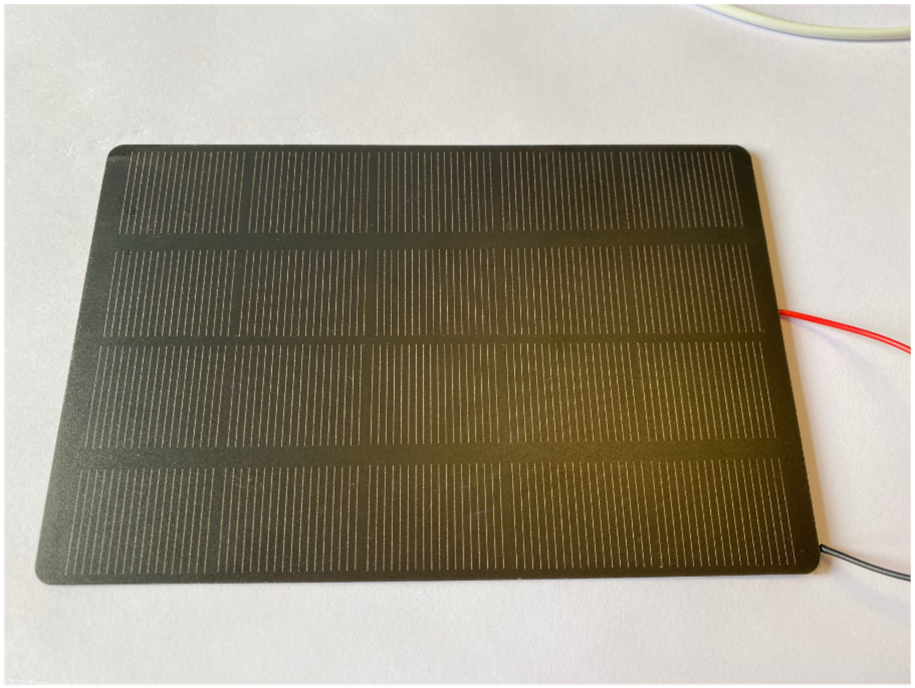


Figure 19: Solar panel

We use a mono-crystalline silicon solar panel with size 153mm x 99mm. We choose to use mono-crystalline silicon solar panel because it has the highest energy efficiency compared with other solar panel types like poly-crystalline silicon solar panel and thin film solar panel. At the condition that light intensity is 1000W per square meter, the solar panel can generate 2.5W output.

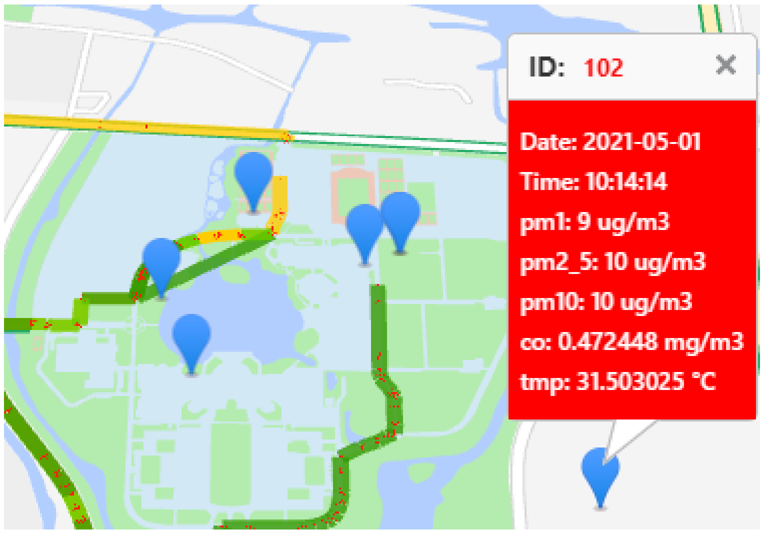
8 Verification

In this section, we will review the requirement and verification of our project from the perspective of software, algorithm and hardware with the focus on mechanical parts and power module.

8.1 Software Verification

8.1.1 Air-quality data update every 10 seconds

The air-quality data shown on the web page will update every 10 seconds inside the red block shown in the figure 20. It indicates that the server can upload the data every 10 seconds.



ID: 102

Date: 2021-05-01
Time: 10:14:14
pm1: 9 ug/m3
pm2_5: 10 ug/m3
pm10: 10 ug/m3
co: 0.472448 mg/m3
tmp: 31.503025 °C

Figure 20: Air-quality shown on the web page

8.1.2 Trajectory Correction of the Map

The collected GPS information may not be exactly on the road and sometimes it is deviated from the road. We used the "trajectory correction" module in the [8] and match the GPS data to its nearest street. In figure 21, we can see the results of the trajectory point after map matching.

8.2 Algorithm Verification

8.2.1 Dataset and Performance Metric

To validate the algorithm in this project, we adopted two open-source data sets to test. The first one is the Binhai dataset[11] in which the air quality of a costal area in Binhai district in Tianjin is measured from July 1st to December 31st, 2018. The second one is the Antwerp dataset [23], located in the cityof Antwerp, Belgium.

We want to take a small portion of it to be as input to train our model due to the limited computational resources. In this project, we use the RSME (Root Square Mean Error) as the evaluation metric. We take 90% of the non-zero entries as the training data and take the remaining 10% as validation.

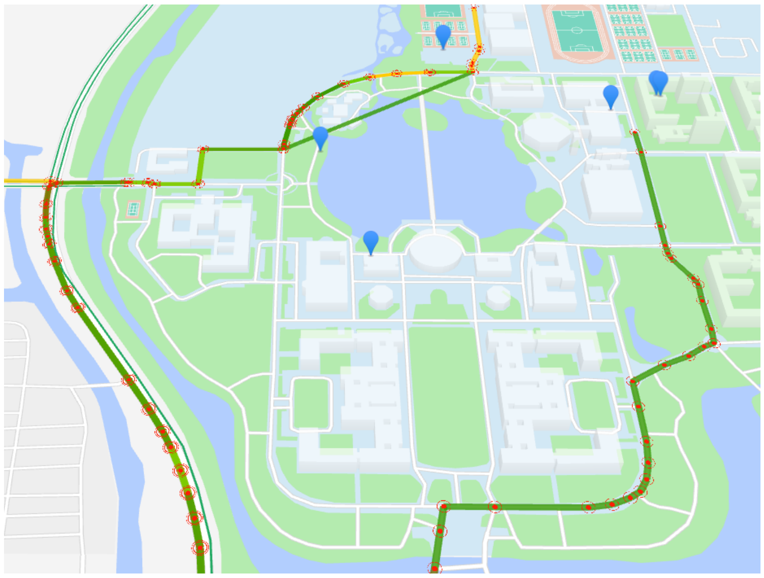


Figure 21: Map matching of our campus

8.2.2 Results

Table 3 shows the RSME value on both data sets using the baseline interpolation method and our kNN-based collaborative filtering method. Also, figure 22 is the output on our website that reflects on the air-quality condition in Haining City in May 2nd.

Method	Binhai Dataset (2018, Aug)	Antwerp Dataset (2019 April)
Linear Interpolation	19.0	4.5
KNN-based collaborative filtering	16.5	5.2

Table 3: Experiment results on the open-source data sets

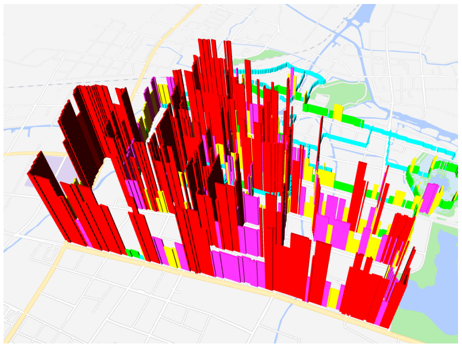


Figure 22: An example of output of inference algorithm in Haining

8.3 Hardware Verification

8.3.1 Mechanical Parts Verification

From figure 23 and figure 24, the installation of our device on a bike is shown. We used the device holder to fix our device on the bike. We have tested its robustness in various road conditions, for example the deceleration strip and it was fixed perfectly on the bike.

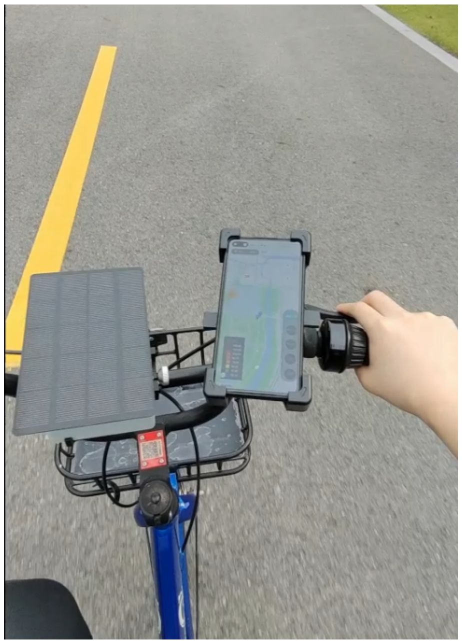


Figure 23: First perspective of the installation of our device

Figure 24: Third perspective of the installation of our device

8.3.2 Power Module Verification

The goal of the power module is to provide power for long-term running of the device. More specifically, When riding, the battery should be charged and the power from the kinetic energy module and solar energy module should cover the device power consumption.

Current Requirement

We measure the current consumption of our final version hardware circuit, the measurement ranged from 0.4A to 0.5A.

Verification of Solar Energy Module

We tested the solar panel on a sunny day. The weather condition is shown in the figure below. We measured the short circuit current and open circuit voltage. Measurements are shown in the table above. If the weather is cloudy, the current provided by solar panel will be very low, on average below 50mA.

Verification of Kinetic Energy Module

Below is the relationship between the riding speed and the charging current provided by kinetic energy module. The common riding speed is in the range of 15km/h to 25km/h. So the current it can provide is around 0.3-0.5A.

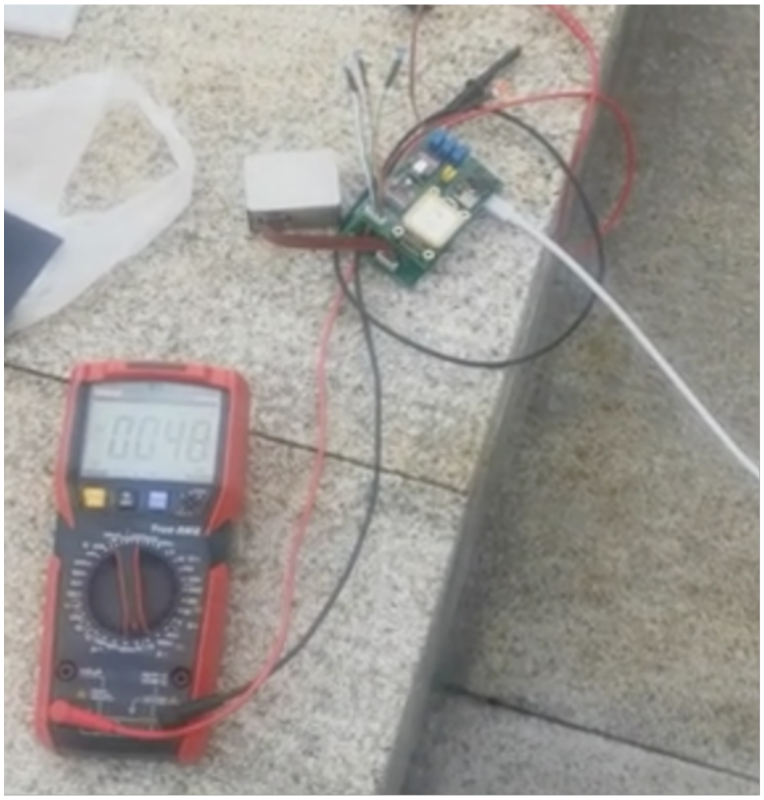
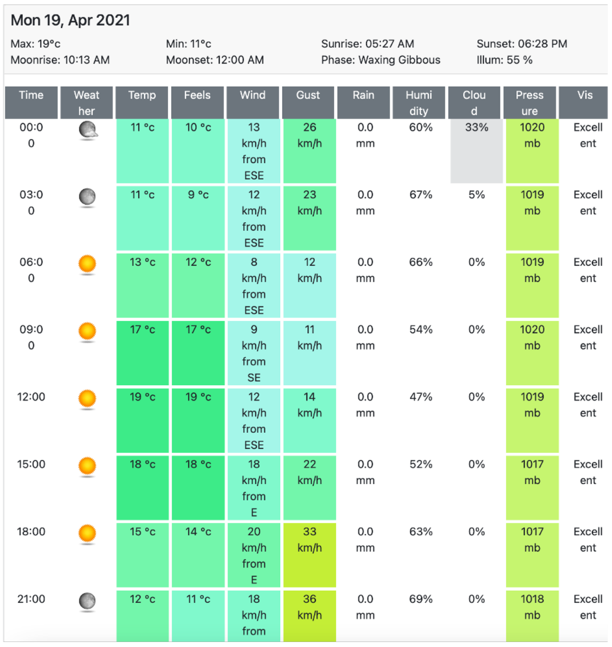


Figure 25: Current required by our device

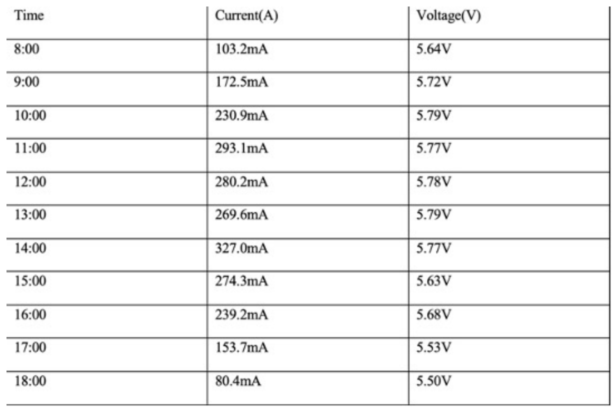


Mon 19, Apr 2021

Max: 19°c	Min: 11°c	Sunrise: 05:27 AM	Sunset: 06:28 PM
Moonrise: 10:13 AM	Moonset: 12:00 AM	Phase: Waxing Gibbous	Illum: 55 %

Time	Weather	Temp	Feels	Wind	Gust	Rain	Humidity	Cloud	Pressure	Vis
00:00	☾	11 °c	10 °c	13 km/h from ESE	26 km/h	0.0 mm	60%	33%	1020 mb	Excellent
03:00	☾	11 °c	9 °c	12 km/h from ESE	23 km/h	0.0 mm	67%	5%	1019 mb	Excellent
06:00	☀	13 °c	12 °c	8 km/h from ESE	12 km/h	0.0 mm	66%	0%	1019 mb	Excellent
09:00	☀	17 °c	17 °c	9 km/h from SE	11 km/h	0.0 mm	54%	0%	1020 mb	Excellent
12:00	☀	19 °c	19 °c	12 km/h from ESE	14 km/h	0.0 mm	47%	0%	1019 mb	Excellent
15:00	☀	18 °c	18 °c	18 km/h from E	22 km/h	0.0 mm	52%	0%	1017 mb	Excellent
18:00	☀	15 °c	14 °c	20 km/h from E	33 km/h	0.0 mm	63%	0%	1017 mb	Excellent
21:00	☾	12 °c	11 °c	18 km/h from	36 km/h	0.0 mm	69%	0%	1018 mb	Excellent

Figure 26: Test condition of solar energy module



Time	Current(A)	Voltage(V)
8:00	103.2mA	5.64V
9:00	172.5mA	5.72V
10:00	230.9mA	5.79V
11:00	293.1mA	5.77V
12:00	280.2mA	5.78V
13:00	269.6mA	5.79V
14:00	327.0mA	5.77V
15:00	274.3mA	5.63V
16:00	239.2mA	5.68V
17:00	153.7mA	5.53V
18:00	80.4mA	5.50V

Figure 27: Test result of the solar panel on sunny day

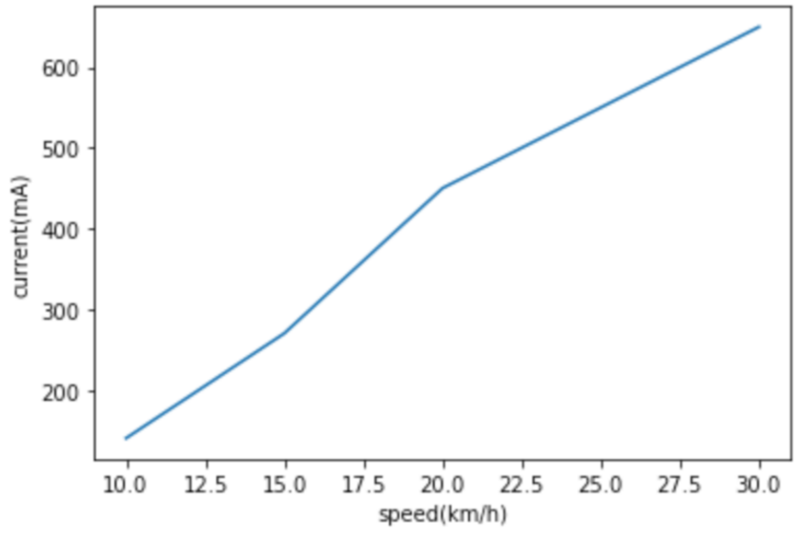


Figure 28: The relationship between speed and charging current

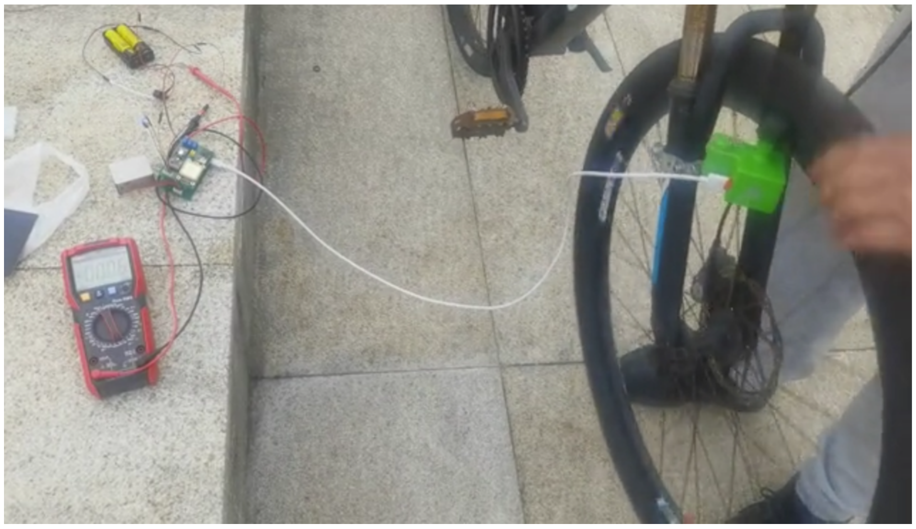


Figure 29: Verification process of kinetic energy module

When we rotate the wheel, the current flow out from the battery is -0.05A, a negative number, which means it is being charged.

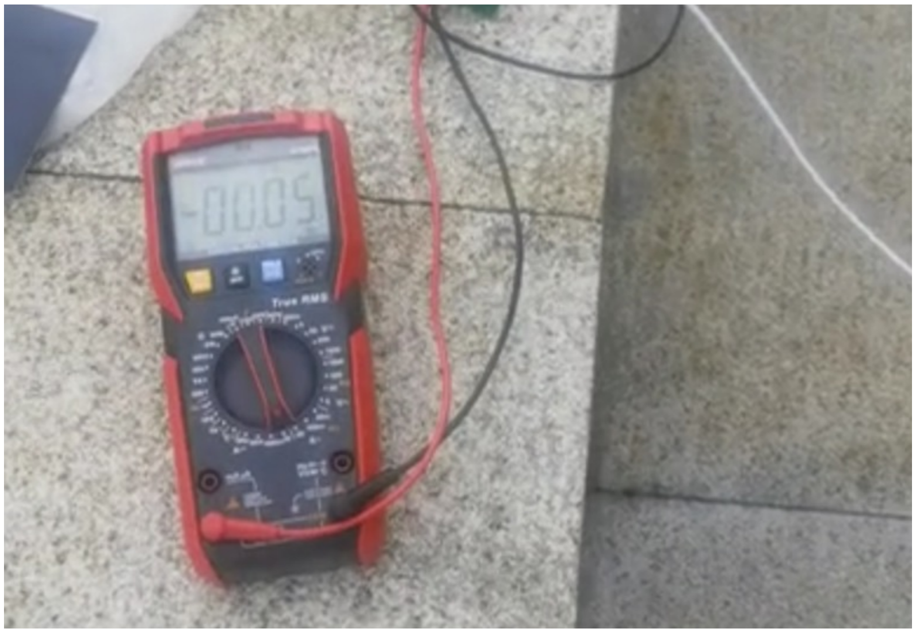


Figure 30: Current of battery when we rotate the wheel

We conclude that our kinetic energy module can provide enough power for our device when riding. Combined with the solar energy module, the whole power module satisfy the requirement of covering the device power consumption.

9 Cost

9.1 Hardware Cost Cost Analysis

Table 4: Parts Costs

Part	Decription	Cost per Unit (RMB)	Number	Others
Stm32F103C8T6	MCU	11.3	1	N/A
ESP01	Wi-Fi Module	16.95	1	N/A
ATGM336H	GPS Module	39.3	1	N/A
PMS5003	PM1, PM2.5, PM10	70	1	N/A
BME680	T, P, Humidity	55	1	N/A
MiCS-6814	CO, NO2, NH3	149.6	1	N/A
Molex 1.25mm 8p connector	N/A	1.08	1	N/A
18650 battery	N/A	45.9	2	N/A
Battery Box	N/A	2.67	1	N/A
Resistor	N/A	0.02	1	N/A
PV panel	10cm*10cm	19.6	1	N/A
PV controller	N/A	5.96	1	N/A
Battery indicator	N/A	5.73	1	N/A
kinetic energy module	N/A	75	1	N/A

9.2 Mechanical Cost Analysis

Table 5: Parts Costs

Part	Decription	Cost per Unit (RMB)	Number	Others
Device Holder	N/A	23	1	N/A
3D Printing Material	Inner Strcture	63.8	N/A	0.5kg
Device Container	Plastic	6.8	1	N/A

9.3 Software Cost Analysis

Table 6: Parts Costs

Part	Decription	Cost per Unit (RMB)	Number	Others
Server Rent	Vlutr	30	1	N/A

9.4 Labor

Table 7: Parts Costs

Part	Decription	Cost per Unit (RMB)	Number	Others
Zhengxin Jiang	150	100	15,000	N/A
Haofan Lu	150	100	15,000	N/A
Haoqiang Zhu	150	100	15,000	N/A
Kaiwen Hong	150	100	15,000	N/A

9.5 Total Cost

Our system consists of large-scale device installed on the public bike system yet at this stage, due to the time and financial limit, we decide only have several samples for functional demonstration. The cost for n unit(s) is 60000 + 589.8n where n is the number of units.

10 Conclusion

10.1 Accomplishments

In this project, we have successfully built a mobile air quality monitoring system for urban sensing. On the hardware side, we have a highly developed design of our mobile air quality monitoring device, which includes the design of the hardware circuit PCB, the stm32 microcontroller programming, the design of multi-source power supply subsystem and the design of the device packaging. The PCB is manufactured by off-campus professional manufacturer and soldered by our team. On the software side, we setup a cloud server for the database system, web server and inference algorithm. We designed the frontend webpage to illustrate real-time air quality around user as well as the air quality map for the entire city. On the inference algorithm side, we implemented an auto-encoder based algorithm and formulated the inference problem as a matrix completion problem. Currently the RSME of our algorithm outperforms the basic algorithm (linear interpolation, Gaussian interpolation and basic LUR model). On the mechanical design part, we bought two commercial products combined with our 3D printing design to manage to build a robust container for the device to be attached on the bike without being damaged by rain.

10.2 Uncertainties

We adopted a deep auto-encoder method for our air quality map inference model, which has great performance but the deep learning method is still a black box for researchers in this field and there are rare principles or solid explanations for this method. Therefore, there are uncertainties of adopting the deep learning based method.

10.3 Ethical considerations

Our design needs to collect the data of location, time, and air quality statistics. Therefore, user privacy is a problem that we need to take into account. ACM ethics code 1.6 requires us to respect privacy [24]. To protect user privacy, we will only collect necessary data (time, location and air quality sensor data) required for air quality map service. User is anonymous to our system whether connecting to the on-bike device or visiting the air quality map website. Our system requires no permission to access user's smartphone to ensure user privacy.

10.4 Future work

For hardware circuit design, currently the sensors of MICS6814 and BME680 we use are the pin header version come with complete peripheral circuit, which is ready to be directly connected to the microcontroller. For a more mature and commercial design, the peripheral circuit can be self-designed and all components can be on one PCB to get a more compact design.

For user interface, a mobile app can be developed instead of the current webpage.

References

[1] K. Aberer, S. Sathe, D. Chakraborty, A. Martinoli, G. Barrenetxea, B. Faltings, and L. Thiele, "Opensense: Open community driven sensing of environment," in *Proceedings of the ACM SIGSPATIAL International Workshop on GeoStreaming*, ser. IWGS '10. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: https://doi.org/10.1145/1878500.1878509 p. 39–42.

[2] J. J. Li, B. Faltings, O. Saukh, D. Hasenfratz, and J. Beutel, "Sensing the air we breathe: The opensense zurich dataset," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, ser. AAAI'12. AAAI Press, 2012, p. 323–325.

[3] Y. Hu, G. Dai, J. Fan, Y. Wu, and H. Zhang, "Blueaer: A fine-grained urban pm2.5 3d monitoring system using mobile sensing," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. IEEE Press, 2016. [Online]. Available: https://doi.org/10.1109/INFOCOM.2016.7524479 p. 1–9.

[4] Y. Cheng, X. Li, Z. Li, S. Jiang, Y. Li, J. Jia, and X. Jiang, "Aircloud: A cloud-based air-quality monitoring system for everyone," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2668332.2668346 p. 251–265.

[5] X. Xu, X. Chen, X. Liu, H. Y. Noh, P. Zhang, and L. Zhang, "Gotcha ii: Deployment of a vehicle-based environmental sensing system: Poster abstract," 11 2016, pp. 376–377.

[6] Y. R. Yuhu Xu, Ke Liu, "Air quality monitoring system based on sensor network," 2020.

[7] I. E. T. F. (IETF), "The geojson format," Internet Requests for Comments, RFC Editor, RFC 7946, 8 2016. [Online]. Available: http://www.rfc-editor.org/info/rfc7946

[8] *Amap JS API v2.0*, Alibaba Group, Amap open platform, Hangzhou, Zhejiang, 2020.

[9] *Loca JS API v2.0 beta*, Alibaba Group, Amap open platform, Hangzhou, Zhejiang, 2020.

[10] R. L. Ott and M. T. Longnecker, "Linear regression and correlation," 2004.

[11] R. Ma, N. Liu, X. Xu, Y. Wang, and L. Zhang, "Fine-grained air pollution inference with mobile sensing systems: A weather-related deep autoencoder model," *Proceedings of the ACM on Interactive Mobile Wearable and Ubiquitous Technologies*, vol. 4, no. 2, pp. 1–21, 2020.

[12] N. Holmes and L. Morawska, "A review of dispersion modelling and its application to the dispersion of particles: An overview of different dispersion models available," *Atmospheric Environment*, vol. 40, no. 30, pp. 5902–5928, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1352231006006339

[13] P. Zannetti, "Air pollution modeling: theories," 1990.

[14] Z. Zlatev and I. Dimov, "Volume 13. computational and numerical challenges in environmental modelling," 2006.

[15] M. Wu, J. Huang, N. Liu, R. Ma, Y. Wang, and L. Zhang, "A hybrid air pollution reconstruction by adaptive interpolation method," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems, SenSys 2018, Shenzhen, China, November 4-7,*

2018, G. S. Ramachandran and B. Krishnamachari, Eds. ACM, 2018. [Online]. Available: https://doi.org/10.1145/3274783.3275207 pp. 408–409.

[16] Y. Hu, J. Fan, H. Zhang, X. Chen, and G. Dai, "An estimated method of urban pm2.5 concentration distribution for a mobile sensing system - sciencedirect," *Pervasive and Mobile Computing*, vol. 25, pp. 88–103, 2016.

[17] C. Yun, X. Li, Z. Li, S. Jiang, and Y. Li†, "Aircloud: a cloud-based air-quality monitoring system for everyone," *ACM*, 2014.

[18] Y. Xu and Y. Zhu, "When remote sensing data meet ubiquitous urban data: Fine-grained air quality inference," in *2016 IEEE International Conference on Big Data (Big Data)*, 2016.

[19] H. Zhong, C. Yin, X. Wu, J. Luo, and J. W. He, "Airrl: A reinforcement learning approach to urban air quality inference," 2020.

[20] Z. Hu, Z. Bai, Y. Yang, Z. Zheng, K. Bian, and L. Song, "UAV aided aerial-ground iot for air quality sensing in smart city: Architecture, technologies, and implementation," *IEEE Netw.*, vol. 33, no. 2, pp. 14–22, 2019. [Online]. Available: https://doi.org/10.1109/MNET.2019.1800214

[21] T. H. Do, E. Tsiligianni, X. Qin, J. Hofman, V. P. La Manna, W. Philips, and N. Deligiannis, "Graph-deep-learning-based inference of fine-grained air quality from mobile iot sensors," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8943–8955, 2020.

[22] "Stm32f103c8t6 - blue pill development board pinout, specs, equivalent datasheet." [Online]. Available: https://components101.com/microcontrollers/stm32f103c8t8-blue-pill-development-board

[23] T. H. Do, E. Tsiligianni, X. Qin, J. Hofman, V. P. La Manna, W. Philips, and N. Deligiannis, "Graph-deep-learning-based inference of fine-grained air quality from mobile iot sensors," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8943–8955, 2020.

[24] A. Machinery, "Acm affirms obligation of computing professionals to use skills for benefit of society," 2018.