Low-Cost High-Precision PM Sensor

Zhaoqi Ma
Electrical and Computer Engineering
Univesity of Alberta Faculty of Engineerring
Edmonton, AB, Canada
zm5@ualberta.ca

Xiaolei Zhang
Electrical and Computer Engineering
Univesity of Alberta Faculty of Engineerring
Edmonton, AB, Canada
xiaolei6@ualberta.ca

Zixuan Wan
Electrical and Computer Engineering
Univesity of Alberta Faculty of Engineerring
Edmonton, AB, Canada
zwan1@ualberta.ca

Siru Chen
Electrical and Computer Engineering
Univesity of Alberta Faculty of Engineerring
Edmonton, AB, Canada
siru2@ualberta.ca

***Abstract*— This paper discusses a practical, low-cost, high-precision particulate matter (PM) sensor based on the light-scattering detection method, as well as an Android application that supports reading Real-time data through BLE and generating graphs for the dataset. A prototype system costing $85.05 was designed, built, and tested for reading, processing, and communicating data. The prototype system comprises a sensor end and a smartphone end. The Arduino platform is the core data processing unit because of its simplicity and abundant resources. Three Sharp light-scattering PM sensors with a DHT temperature humidity sensor were deployed to enhance the precision and reduce part-to-part variation in the hardware level. A 2-stage filtering algorithm was used to remove short-term noise[2], and a multiple linear regression Tensorflow module was developed to calibrate the PM sensors[1] with a commercial laser PM sensor. However, the trained Tensorflow model does not perform well, and Arduino's computation power and memory resources cannot support the complex Tensorflow model. The final performance of the three-sensor fusion solution is not satisfactory. So, the final prototype's sensing system is based on one sharp light-scattering PM sensor with a DHT11 temperature and humidity sensor. The system was successfully built and tested to be meeting our initial requirements.**

Keywords – Low-cost PM sensor, High-precision PM sensor, PM sensor fusion, Linear Regression module

I. INTRODUCTION

The health concerns created by air pollution have mainly been reported and promoted by governments worldwide for the past few decades. While air pollution is composed of numerous pollutants, the most harmful and insidious one is particulate matter(PM)[3]. Particulate matters are tiny particles with diameters below 1um[4] that can penetrate the human respiratory system and reach the bloodstream, potentially causing severe health problems, including respiratory diseases, cardiovascular diseases, asthma, and lung cancer [3]. Air pollution had resulted in over 8million [2][5] deaths and 9million[3][6] deaths in 2014 and 2019, respectively. Therefore, controlling air pollution is urgent and crucial. The first step to controlling pollution is measuring and monitoring PM density, the measurement of overall density (in terms of mass) of PM within one cubic meter; thus, the unit of PM density is ug/m3. According to the Department of Health of the New York State Government, the standard safe PM density for the short term (24h average) is 35 ug/m3 and for the long term (annual average) is 12 ug/m3[7].

Air pollution is omnipresent, meaning both indoor and outdoor air pollution are present. This makes PM monitoring at an average citizen's home equally important as PM monitoring at governmental and research facilities that endeavour to control outdoor air pollution at a macro level. Furthermore, it is much easier to decrease indoor pollution than outdoor pollution. For instance, one with moderate wealth can purchase an air purifier after knowing that the air quality at their home is beyond the safety range. At the same time, low-income families can circulate the indoor air more frequently by having their windows opened for a period. However, due to inadequate propaganda and costly PM sensors, public awareness of indoor air pollution is lower than outdoor pollution. One way to alleviate the issue is to bring down the cost of commercial PM sensors. Cheap sensor products cost around $30-40 (on Taobao), yet, they are generally inaccurate and cheaply made. Our goal is to make a PM sensing system that costs less than $30 while having accurate performances comparable to multiple times more expensive systems. With the prototype tested successfully, our next goal would be to optimize the mass production system and further put the mass-produced devices into the market. Having this system commercialized will benefit three groups of people:

1. The general public can purchase high-precision sensors at much lower costs, helping them accurately determine their homes' air quality.
2. Researchers who require high-precision sensors in large quantities (to build a sensor array) can accomplish their projects at a much lower cost.
3. If the market for our device is anticipated to be large, commercial companies producing and selling our device may gain a considerable amount of profit.

Traditional reference PM sensing methods like the gravimetric method use filtering paper to collect air particles and then measure the particle density of the collected sample [2][4]. Cheaper methods like the Beta-ray based measurement also require a filtering paper, and the measurement precision is well below the gravimetric method [2]. The most common low-cost sensing method that does not require a filtering paper is the light-scattering method, which measures the intensity of refracted light by PM particles in a sampling hole [2][4]. Several successful low-cost, high-precision PM sensors based on the light-scattering method have been demonstrated, and two systems caught our attention. The first system is presented by Cho and Beak [2], which uses a Sharp GP2Y1010 IR light-scattering PM sensor along with a temperature-humidity sensor for detection and an MCU for data processing. Also, an Android

application could be connected to the sensor using Bluetooth Low Energy(BLE). The system used a filtering system and a calibration algorithm to enhance measurement accuracy. The temperature & humidity sensors were used to offset reading variations caused by temperature and humidity changes. The second system is presented by Ordinas et al; they used the cheap Ozone and Nitrogen dioxide sensors array to reduce part-to-part variation and deployed a multivariable linear regression module to calibrate and fuse the sensor array with reference to several air quality stations. However, their system did not use a filtering algorithm to reduce noise.

Our design is based on the designs mentioned above, but we aimed to develop a better design that merges the merits of the above two. Our design comprises a sensor end along with a smartphone end. The two can connect via BLE for real-time reading. Historical data can also be plotted by transferring a .csv file from the sensor unit to the smartphone end using a micro-SD card. The sensor unit comprises three PM sensors with a temperature-humidity sensor, and the sensor inputs will be processed by a filtering algorithm and machine learning for real-time calibration.

The paper's outline is as follows—section II Prototype Design, Implementation, and Results; Section III Future work; Section IV Conclusion.

II. PROTOTYPE, DESIGN, IMPLEMETATION, RESULTS

A. *Prototype Design*

1) *Design Principle:*

The core design philosophy of the low-cost, high-precision is to calibrate cheap sensors to the extent that it is as accurate as much more expensive sensors by using software algorithms and machine learning. The initial plan is to use three PM sensors and a temperature-humidity sensor to offset the part-to-part variation and environmental changes. Cheap light-scattering PM sensors are mass-produced and uncalibrated so that each sensor will have differences. The light-scattering PM sensor has three functional components: one infrared LED, one photodiode receptor, and a voltage amplifier. The sensor senses current PM values by measuring the voltage generated in the photodiode receptor when the PM particles refract infrared light from the LED. The amount of light refracted is positively correlated with the particle density, and as a result, the output voltage from the photodiode is linearly proportional to the PM density. Since a photodiode is used, dark current will inevitably influence the output voltage, which is positively correlated with temperature. Furthermore, humidity also affects the accuracy of PM reading, so it is necessary to offset the variations created by temperature and humidity differences (to a reference temperature and humidity at which we calibrated our device). The final important consideration point is the possibility of future commercialization, so it is necessary to develop an Android application with excellent functionality and aesthetics that would make it possible to read real-time and historical data from the sensors.

2) *Key requirements:*

1. Sensor system should be battery powered and should be able to run for 30 days without battery replacement.
2. The accuracy of the sensor system should be higher than the original non calibrated PM sensor.
3. Sensor system should have a Bluetooth connection to communicate with a smartphone.
4. An Android App should be developed to communicate with the sensor end and show the real-time PM data to users.
5. Sensor system should be able to store measured data.
6. Sensor systems should have data processing and filter algorithms to improve accuracy.
7. Sensor system should be portable.

3) *Prototype:*

Our prototype consisted of one PM sensor, one temperature & humidity sensor, one Arduino MKR WIFI 1010 as the microcontroller unit with Bluetooth low energy chip build inside, one SD card module , one Micro 2GB SD card, one real-time clock (RTC) module, one 3.7V to 5V voltage regulator, and one 3.7V 2000mAh Li battery The PM and temperature humidity sensors were responsible for sensing environmental PM (by measuring receptor voltage in mV), temperature (measured in Celsius), and humidity (measured in %relative humidity (RH)) respectively; the microcontroller unit is responsible for reading and processing signals from the sensors, serializing the processed data for Bluetooth transmission, and storing processed data into the SD card; the Bluetooth module is used to transmit and receive data from the Android application; the SD card module is used to interface the Micro SD card with the microcontroller; the RTC is used to keep track of real-world time when the sensor is disconnected; the battery is used to power the whole sensor system. Except for the voltage regulator, all hardware listed above was connected to the microcontroller. The voltage regulator would allow the battery to power up both the microcontroller and the sensors.

After the microcontroller reads the raw signals, a two-stage filtering algorithm will act to filter out the spike noise. The first stage is a median filter that groups five input values into one sorted array for each sensor and then chooses the median value as the selected input value to be passed to the second stage. The second stage was a moving average filter that autocorrected the input value based on the average value of the previous nine values. For calibration, a simple linear regression Tensorflow module was applied to linearly relate the filtered voltages with the PM densities sensed by the 352® M25 sensors. The linear regression model was

then loaded into the microcontroller, and every filtered PM value will be fed to the model, and the model will predict a PM value as the output. Then the output value from the model will be compensated by temperature and humidity values. Due to the limited time, we did not have the time to test variations caused by temperature and humidity differences ourselves. Hence, we calculated the relevant constants from the data given by Cho and Y. Baek [2]. The final model that converted every filtered input signal into a PM value is shown in the (1):

$$PM\ density\ \left(\frac{ug}{m^3}\right) = \alpha(V - \beta\Delta T - \gamma\Delta H) + \delta \quad (1)$$

Where α is the constant term obtained from machine learning; β is the temperature offset constant calculated based on the value from Cho and Y. Baek [2]; ΔT is the temperature difference to the reference temperature; γ is the humidity offset constant calculated based on the data from Cho and Y. Baek [2]; ΔH is the humidity difference to the reference humidity; δ is intercept value contained from machine learning model.

The converted PM density data would be stored on the SD card in a .csv file with a timestamp given by the RTC module, no matter whether the Bluetooth is connected. Real-time reading could be done by connecting with the Android app. The application also allows a graphical view of the .csv file loaded from the SD card for historical data, and the Android App will upload all read data to the Firebase cloud server.

B. Implementation

1) Main development task:

The prototype was implemented using one Sharp GP2Y1026 digital PM sensor, an Adafruit DHT11 temperature humidity sensor, an Arduino MKR WIFI 1010 for the microcontroller, a u-blox NINA W-102 Bluetooth module, a 2GB micro-SD card with an SD card module, a real-time clock, and a Xiaomi 1000mAh power bank. The sharp GP2Y1026 sensor was chosen due to its stability, low price, and built-in analog to digital converter. This sensor directly outputs a stream of digital signals representing the sensed voltage for each reading through a UART interface to Arduino. Adafruit DHT11 was chosen for its low price and the available Arduino libraries. The NINA-W102 Bluetooth low energy module was integrated into the Arduino board, making us choose the Arduino MKR WIFI 1010 evaluation board. The Xiaomi power bank was used instead of a smaller lithium battery because we could not find a battery with the compatible JST connector in China, so our implementation did not use the voltage regulator. A detailed hardware architecture is shown in Fig.1. The physical device that we built is shown in Fig.2.

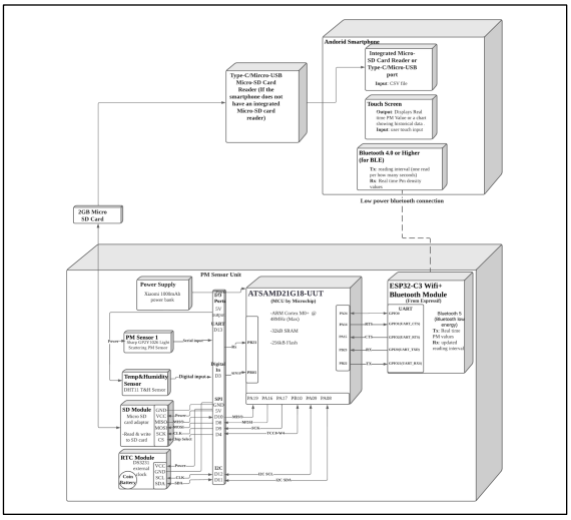

Fig. 1. System architechure of the prototype

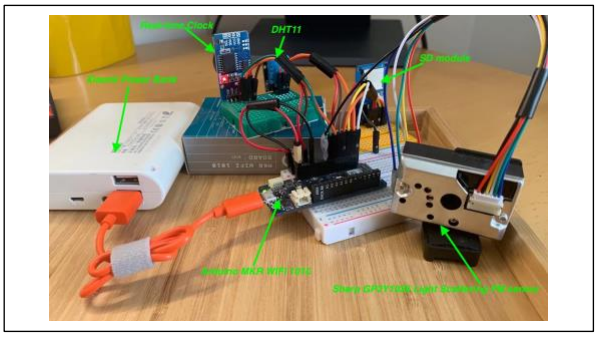

Fig. 2. Photo of the physical hardware device

Firmware was written using C++, and the development and testing were done using Visual Studio Code. The Android software application was developed and tested using Android Studio

2) Timeline:

The developers were divided into two groups, one group was focused on Android App development, and the other group focused on the firmware and hardware development on Arduino. The first stage was focused on the minimum design, which was completed at the end of February. This part included the successful reading of a PM sensor and a temperature and humidity sensor, the successful BLE communication between the smartphone and the Arduino end, the successful storage of the SD card, and the successful configuration of the RTC module. The second phase was completed at the beginning of April. The smartphone side mainly developed the data storage based on Firebase, the creation and saving of user accounts, the graphing function of .csv files, and the optimization of UI; the Arduino development side actively tried to use three PM sensors and the temperature and humidity sensors for

fusion and calibration and trained the Tensorflow model. The third phase was completed on April 5, and the developer system was tested entirely.

C. Result

Due to several hardware technical difficulties, firstly, even if we used an expensive voltage signal source to power up the sensor array and use that source as ADC reference voltage, the original Sharp GP2Y1014 PM sensor is not sensitive and stable enough. Secondly, we tried to use the SAMYOUNG DSM501 dust sensor [10], but the digital signal output of DSM501 was disrupted when we powered up the BLE chip on Arduino. So, we changed the sensor unit to Sharp GP2Y1026[9], which utilized the UART interface for communicating with the microcontroller. However, only one UART port in our microcontroller was left for peripheral connection, so only one Sharp GP2Y1026 PM sensor was used instead of three.

By using the two-stage filtering algorithm, the input signals became significantly stable. However, this would lead to computation latency and a decrease in the short-time sensitivity. The computation latency was not noticeable if only one PM sensor was used, but the latency is obvious if we used three sensors. However, reducing short-time sensitivity was not a significant issue for our application since rapid PM changes are rare in natural environments. The first stage median filter grouped five raw signals into an array and selected the median. In the second stage, moving the average filter will autocorrect the input signal based on the average value of the previous nine values. The first nine values would be passed to the moving average filter. Fig.3 shows the filtering effect in a stable room environment.

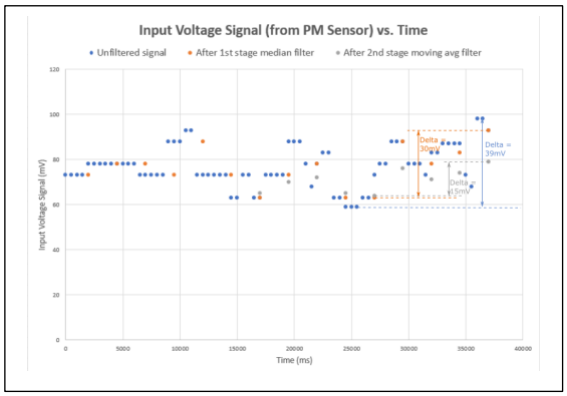

Fig. 3. Comparision between raw signal and filter signals at different stages

As Fig.3 shows, the difference between the biggest and smallest raw signals is 39mV. Nevertheless, this value is reduced to 30mV after first-stage filtering and 15mV after complete filtering. A low value is desired as the sensing was conducted in a stable room environment.

A linear machine learning process was applied to the fully filtered PM voltage to linearly relate it to the PM values detected through the 352® M25 Laser PM2.5 detector at 17°C and 45%RH. Data collection for machine learning was done in two environments, one in a clean air room and the other in a closed room with burning incense. The PM value detected from M25 was recorded at a very close time when an output voltage from the Sharp sensor was read. A comparison between the converted PM density using the conversion algorithm provided by the Sharp GP2Y1026 datasheet and 352® M25 Laser PM2.5 detector sensed PM density is shown in Fig.4. The discrepancy between the two values was significant, illustrating the necessity of calibration. The linear model generated from machine learning is shown in Fig.5, α and β constants were $0.3653\frac{ug}{m^3 mV}$ and $5.5429\frac{ug}{m^3}$, respectively. Fig.6 directly compares the results before and after machine learning. The machine-learned linear model exhibited a noticeably lower discrepancy with the measured data from M25 compared with Sharp's simple conversion algorithm.

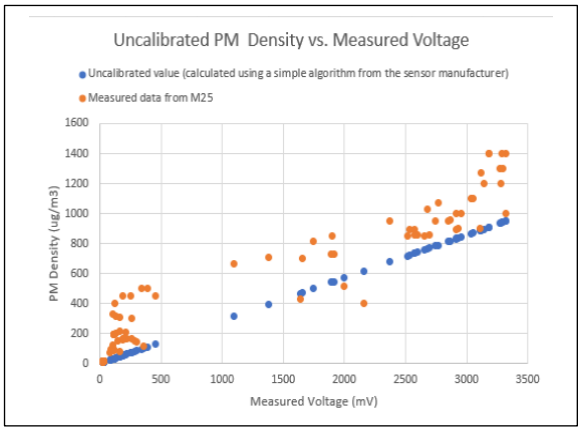

Fig. 4. Comparision between calcualted PM using Sharp datasheet and measured PM value from M25

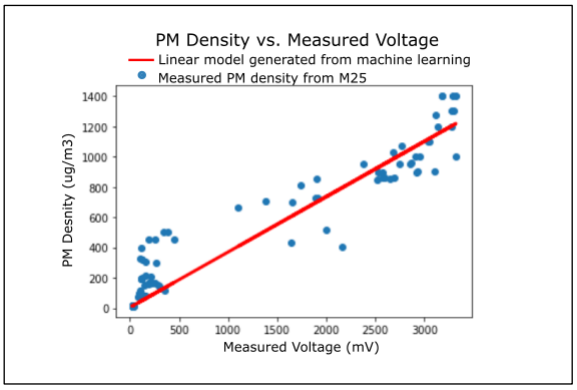

Fig. 5. Comparision between linear fit generated by linear regression module and the measured PM value by M25

Before generating actual PM density, the last step was to compensate for the variation caused by temperature and humidity differences. Due to our project's limited time and testing environment, we could not measure the voltage change against temperature and humidity changes. Therefore, the offset constants were calculated from Cho and Y.Baek's work [2]. The temperature offset constant was obtained from the gradient of

voltage change vs. temperature change. The voltage change from 2℃ to 45℃ was reported to be 129mV, and by assuming it is a linear model, the temperature offset constant was calculated to be around 3mV/℃. The humidity offset constant was directly borrowed from their results which is 0.64mV/%RH. Fig.7 shows the effect of the offset algorithm. Reference temperature and humidity were the environmental conditions (17℃ and 45%RH) during the data collection for machine learning. With all the constants known, (2) was used as our calibrated conversion algorithm to merge all input signals into a physical PM density.

$$PM\ Density\left(\frac{ug}{m^3}\right) = 0.3653(V - 3(T - 17℃ - \gamma(RH - 45\%)) + 5.5429 \quad (2)$$

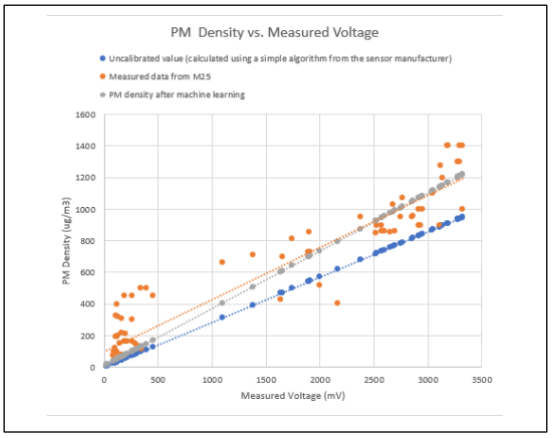

Fig. 6. Comparision between the raw conversion algorithm, the machine-learned model, and the measured PM value by M25

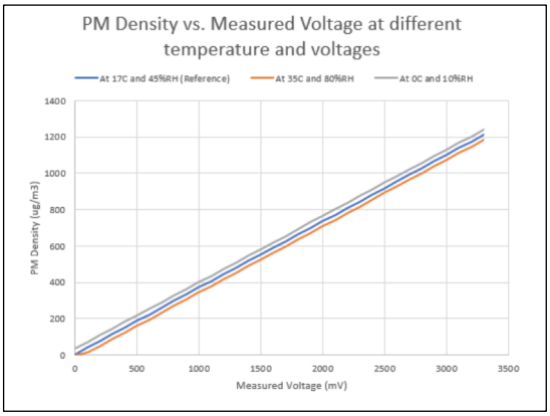

Fig. 7. Comparision between the generated PM values at different environment

Other than the effective calibration, sending the data through Bluetooth and storing the data into SD card were also finished along with the completion of the Android application.

The Android application uses the Bluetooth interface and BluetoothGatt related class provided by Android API and successfully built a BLE connection between the user's smartphone and hardware that can capture real-time data [8]. The Hardware team first chooses a specified service UUID representing their hardware device, three specified characteristic UUIDs representing three values that need to be transferred: current times, user-defined frequency, and PM value. Besides receiving real-time PM values, the user's smartphone devices send current times to the hardware to set the RTC and send a user-defined frequency to the hardware device that can adjust the frequency setting. The application first scans for hardware with predefined service UUID, and once a service with predefined service UUID is discovered, the application will connect the user's smartphone device and hardware device. Later, as a characteristic read operation is successful, the application will receive the PM value sent from the sensor end and present it to our user. Fig.8 shows the BLE connection outcome, the left image shows the real-time data reading, and the right image shows the BLE scanning results where the PM sensor is our hardware device.

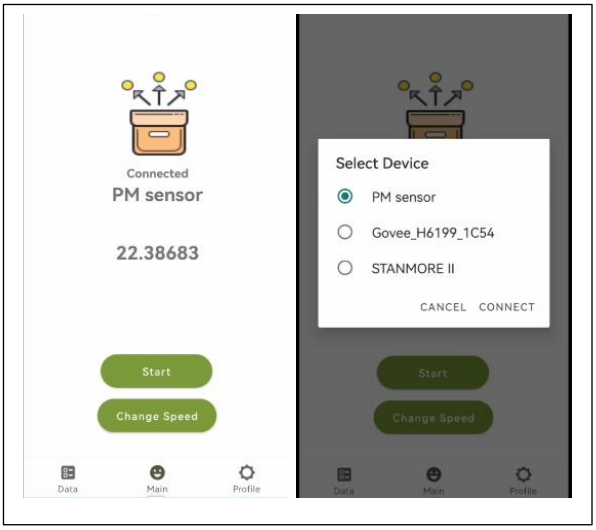

Fig. 8. Screenshot of Andoird App that receive and present PM value via BLE connection

As the real-time data is received from the sensor, if the user is logged in with a specific user account, the system will continue to store and update the collected data and other necessary information into a class and send the entire class to the Firebase Realtime Database. The Firebase Realtime database structure is shown in Fig.9, and a single dataset will be stored under the user's uid branch, then the "Dataset" branch, then the current date branch and then different datasets day can be stored under this specific branch. Thus, for users to get the data from the Firebase, the system will iterate the branches under the user's uid and have a page to show the respective dates, which can be clicked and then show datasets. Users can then click on the specific dataset, and a graph will be generated with Y-axis as PM values, and X-axis is time in seconds. Fig.10

shows the corresponding View-Data page and graph generated by clicking on one of the datasets.

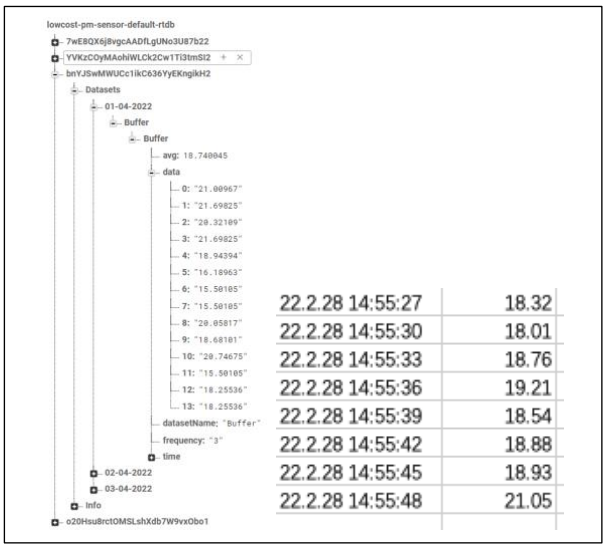

Fig. 9. Data storage Left) On Firebase, Right) On SD card

Fig. 10. Screenshot for Real-time plot features Left) select dataset, Right) genearated PM graph

To plot the diagram from the .csv file stored in the local SD card. The application requires the user to manually transfer the .csv file from the SD card to the smartphone's local storage. After that, the user can then browse through the phone's storage to find the appropriate .csv file to load, and the application would do filtering to output only .csv files. After the .csv file has been selected, the system will automatically generate a diagram representing the PM values in a time domain. The y-axis represents the PM value, and since there would be multiple data, the x-axis would represent the corresponding indices of those PM values. However, the graph would also provide users with a time range indicating the start and end time of the dataset so that the user can then have a better understanding of this dataset. Fig.11 shows how the graph is generated after loading the .csv file.

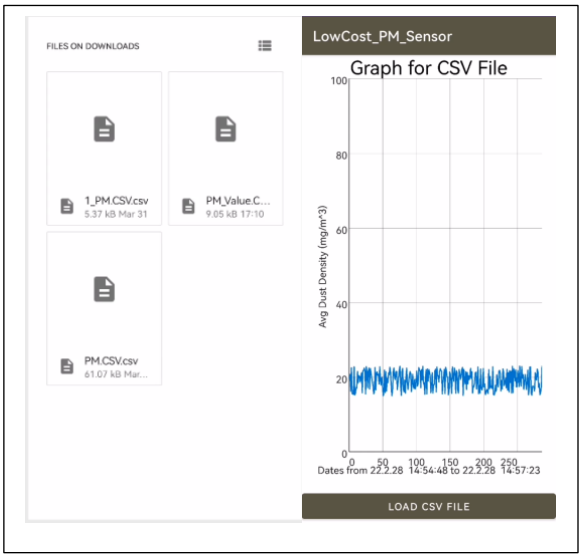

Fig. 11. Screenshot for hisitorical data plot features Left) select .csv file, Right) PM generated graph

III. Future Work

Future work can be done based on the prototype in the following aspects:

More sensor units can be added to the system for better results, but because of the hardware resource limitations of the Arduino, subsequent work should be done on a more powerful piece of equipment. There are three reasons for this. First, The ADC output and digital input become unstable when the Bluetooth chip of Arduino MKR starts working. Second, the on-chip resources of Arduino MKR are limited. We tried to port a slightly more complex multivariate linear model to the development board, but the model needs more power and more memory, and Arduino cannot meet this demand. Third, the Arduino MKR is a single-core Arm architecture, and the Arduino MKR will noticeably feel the latency and lag when calculating large amounts of data. So, if want to improve the computation speed and real-time, better computation modules should be used.

To train a better multiple linear regression model, a suitable dataset of the ground truth should be collected. We used a commercially available laser sensor (352® M25 Laser PM2.5 detector) as our reference during the data set collecting process, but that laser sensor is oversensitive that PM values jumped drastically, so it was difficult for us to capture the actual PM values in relative time. Thus, our dataset for multi-sensor fusion training is not good enough, which led to the subsequent poor linear fit results.

A better filter can be developed. We have combined median filtering and moving average filtering methods.

Although such a filtering method has a good filtering effect on noise, it is not suitable for scenarios where the amount to be measured changes rapidly and is not suitable for impulsive interference scenarios. Therefore, if developers want to improve the reading frequency in the subsequent work, the algorithm can be upgraded.

Add WIFI interface for data transfer (Addition FR & User experience), The prototype design relies on plugging and unplugging SD cards for data transfer, which is inconvenient for users. So in order to improve the user experience, data transfer using WIFI can be developed and used.

Customize a PCB, which can reduce the wiring and assembly errors, facilitate miniaturization of the device, standardize the wiring, improve labour productivity, and reduce the cost of the equipment.

Development of software for IOS, as in the prototyping, was done only on the Android platform.

The temperature and humidity sensor could be updated to DHT22, which provides a broader temperature detection range (-40 to 80°C).

IV. Conclusions

Because of the instability and inaccuracy of traditional light scattering PM sensors, the calibration and training processes need to pay much effort and time. To collect accurate reference data, the experimenter needs to make efforts to control environmental variables as stable as possible. To improve the performance of the training dataset, data cleaning is the crucial process that the experimenter needs to select high-quality data and abnormal data according to experience. Finally, the performance of the filtering algorithm will significantly affect the accuracy of the collected data, thus affecting the performance of machine learning. Moreover, some hardware and firmware problems are interdependent, so the experimenter needs to carefully figure out the possible influence of the firmware code on the hardware. All in all, a low-cost, high-precision PM sensor is built based on the linear regression machine learning algorithm, sensor fusion algorithm, and two-stage filter algorithm. The prototype's traditional cheap light-scattering PM sensor, after calibration and tuning, is as accurate as high-end laser PM sensors. The product chain based on this prototype has also been successfully built, including communication with smart terminals and excellent end-user interfaces.

Acknowledgment

This work was funded by University of Alberta, Dr. Zhenyu Zhang and Dr. Steven Knudsen, under the ECE492 Capstone Project LCHPPMS.

References

[1] J. M. Barcelo-Ordinas, J. Garcia-Vidal, M. Doudou, S. Rodrigo-Muñoz and A. Cerezo-Llavero, "Calibrating low-cost air quality sensors using multiple arrays of sensors," *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1-6, doi: 10.1109/WCNC.2018.8377051.

[2] Cho and Y. Baek, "Practical Particulate Matter Sensing and Accurate Calibration System Using Low-Cost Commercial Sensors," *Sensors*, vol. 21, no. 18, p. 6162, Sep. 2021, doi: 10.3390/s21186162.

[3] I. Manisalidis, E. Stavropoulou, A. Stavropoulos, and E. Bezirtzoglou, "Environmental and health impacts of Air Pollution: A Review," *Frontiers*, 01-Jan-1AD. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpubh.2020.00014/full. [Accessed: 05-Apr-2022].

[4] B. Alfano, L. Barretta, A. Del Giudice, S. De Vito, G. Di Francia, E. Esposito, F. Formisano, E. Massera, M. L. Miglietta, and T. Polichetti, "A review of low-cost particulate matter sensors from the developers' perspectives," *Sensors (Basel, Switzerland)*, 29-Nov-2020. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7730878/. [Accessed: 05-Apr-2022].

[5] WTO, "7 million premature deaths annually linked to air pollution," *World Health Organization*. [Online]. Available: https://www.who.int/news/item/25-03-2014-7-million-premature-deaths-annually-linked-to-air-pollution. [Accessed: 05-Apr-2022].

[6] WTO, "Air Pollution," *World Health Organization*. [Online]. Available: https://www.who.int/health-topics/air-pollution#tab=tab_1. [Accessed: 05-Apr-2022].

[7] "Department of Health," *Fine Particles (PM 2.5) Questions and Answers*. [Online]. Available: https://www.health.ny.gov/environmental/indoors/air/pmq_a.htm. [Accessed: 05-Apr-2022].

[8] "Android.bluetooth : android developers," *Android Developers*. [Online]. Available: https://developer.android.com/reference/android/bluetooth/package-summary. [Accessed: 06-Apr-2022].

[9] Sharp Corporation, "Application note of Sharp Dust Sensor GP2Y1010AU0F," *Sharp GP2Y1026AU0F Dust Sensor*. [Online]. Available: https://global.sharp/products/device/lineup/data/pdf/datasheet/gp2y1010au_appl_e.pdf. [Accessed: 08-Apr-2022].

[10] SAMYOUNG S&C, "Particle/Dust Sensor Module DSM501 Series Specifications." [Online]. Available: http://www.haoyuelectronics.com/Attachment/DSM501A/DSM501.pdf. [Accessed: 08-Apr-2022].