Low-cost Indoor Air Quality (IAQ) Platform for Healthier Classrooms in New Zealand: Engineering Issues

No Author Given *No Affiliation Given*

***Abstract* - Providing a good quality classroom environment where children can breathe in fresh air is important. However, investigating the Indoor Air Quality (IAQ) in large numbers of classrooms is often too costly because currently available commercial brands are too expensive for the majority of schools. We have been developing a low-cost Indoor Air Quality (IAQ) platform called SKOMOBO which can monitor important IAQ parameters such as classroom temperature, relative humidly, particular matter and carbon dioxide level. Because our platform is designed in-house and utilizes low-cost sensors, there is a significant cost reduction and is affordable. In this paper, we discuss the design and implementation of SKOMOBO with the focus in several hardware and software engineering issues to explore the right set of strategies for developing a practical system. Through extensive experiments and evaluation, we have determined the various characteristic and issues associated with developing a low-cost sensor platform and their practical implications and mitigations.**

Index Terms - Indoor Air Quality (IAQ), environmental sensors, sensor monitoring, hardware and software engineering

I. INTRODUCTION

Children spend the second-largest proportion of their time at school. This illustrates that providing a good quality environment, where children can breathe in fresh air in their classrooms, is essential for their health and well-being. A low ventilation rate can lead to a build-up of pollutants, moisture, mould and bacteria that could impact on children's health and performance. International studies have demonstrated links between low ventilation rates and lower children's performance and attention and similarly between low ventilation rates and higher absenteeism [1][2][3].

However, providing adequate ventilation to expel chemical pollutants, moisture and bacteria is very challenging due to the high number of occupants in a classroom. More importantly, investigating the indoor air quality (IAQ) in large numbers of classrooms can be very costly. Commercial available IAQ monitoring equipment has a high capital cost (up to a few thousands of dollars). These issues are typically not achievable for many budget strapped schools.

Fortunately, developments in low-cost sensor R&D, remote logging and data transmission technologies have opened up new opportunities for IAQ monitoring and understanding our indoor environment. For a fraction of the usual cost involved in purchasing IAQ equipment, it is now possible to study indoor climate in each New Zealand classroom and ensure that children are learning in healthy environments.

With the support of the Building Research Levy, a team of researchers at Massey University, in collaboration with NIWA New Zealand, has been developing a low-cost monitoring suite called SKOol MOnitoring BOx (SKOMOBO). A small box, approximately the size of 100 * 100 * 100 mm, was designed to house some low-cost sensors. SKOMOBO is designed in such a way which monitors main IAQ parameters such as classroom temperature, relative humidity, particular matter and carbon dioxide. The full set of sensors in SKOMOBO is expected to be an essential tool for documenting our classroom environment.

The efforts towards developing a low-cost IAQ platform are nothing new and are easy to find from recent studies. However, most of these studies focus on algorithms to calibrate reading of different sensors [4][9][10], or technologies such as system architecture [13], data collection [1][2] and networking techniques [5][8]. Or else the focus of the study tends to target for large scale networks such as smart city initiatives [11][12] or urban planning [6][7]. There are not enough studies in the literature to deal with practical engineering issues in real world applications and often do not help to figure out issues relate to developing sensor technologies for classroom environment.

In this paper, we present the design and implementation of SKOMOBO that integrates a number of low-cost environmental sensors plugged into a commonly used Arduino board. SKOMOBO provides a number of IAQ information by collecting independent sensing information from various air quality components and forwarding the collected data to the remote server. During the development phase of our system, we faced many practical engineering implications relate to both hardware and software. We present some of the design and implementation choices one should consider to choose the right set of strategies in order to develop a successful low-cost sensor platform for classroom environment.

The paper is organised as following: Design considerations are presented in Section II. Section III discusses hardware issues and mitigation techniques. Section IV examines practical software issues in depth and mitigation approaches. Section V illustrates experimental results to show the data reading from SKOMOBO is accurate and reliable. Section VI concludes the paper along with the future work.

II. Design Goals

Designing a low-cost IAQ platform focusing on classroom environment is a complicated and challenging task because there are many factors to consider. Here, we discuss a number of priorities therefore affected the design and the direction of implementation for the SKOMOBO.

A. *Low-cost*

Unfortunately, most commercial IAQ monitors cost upwards from $2,500. From the inception of the project, we realised that most public schools could not afford buying such costly products to understand the classroom environment their students sit in. One goal that was clear to us were to investigate a novel sensor platform that would cost significantly less than what was commercially available so that more schools (e.g., locates in low socioeconomic regions) can deploy air quality monitoring tools for the classrooms to measure and learn about better ventilation practices. We also wanted to measure a number of different air quality related environmental sensors to examine the multitude of different properties including CO_2 amount, dust levels, temperature and humidity, and their relation with occupancy. We also wanted the system to record the times that each round of measurements is taken in order to track the changes in the environment over time.

B. *Reliability*

In order for the measurements to be trustworthy the system must also record its data with the highest accuracy possible, with minimal data loss, and record the data in the correct format. We decided to use two storage systems to store all our sensor data. We use a SD card installed in the Arduino board as a primary storage. After saving to the primary storage, the data is sent to a remotely hosted database (e.g., at Massey University) as a secondary storage which is also backed up in regular interval. These two storage approaches is designed to ensure data safety as well as reliability in case any data loss due to faulty hardware or a change in the environment or system.

C. *Deployment and Maintenance*

The system must be easy to program new units and deployment teams need to be able to deploy updates and patches with minimal difficulty. Existing sensor platforms often require a direct access to the sensors for any upgrades (or if patches are needed to apply at a later stage) by a deploy team. Such travel typically attributes the rise of overall cost. To reduce such hassle, our system offers a remote installation for updates and patches without deployment team having the access to SKOMOBO box.

It must also be easy to verify whether SKOMOBO is running successful and that is running without any errors. It must be easy to notice when a fault or error occurs in the system. SKOMOBO is designed in such a way that it alerts to a system administrator when an error occurs (e.g., the box is down due to power failure or some data can't be read for some intervals). The alert message contains most likely cause of the error.

D. *Privacy*

All sensitive data acquired from the schools and institutions that are being studied must be transmitted and stored in a secure manner to protect their privacy and to prevent tampering by malicious third parties. The most widely used technology to protect privacy is often encryption. But encryption isn't something applicable for many sensor systems due to its tiny OS and memory size because encryption demands a heavy cryptographic computation. Instead, we use a firewall technique to send data only between designated clients (SKOMOBOs) and the server (Massey server). SKOMOBO boxes use anonymous IP addresses to send data to the server whose IP addresses are only known to the server therefore cannot be intercepted by malicious third parties.

III. Hardware Engineering

A. *Circuit Board Design and Microcontroller*

Since our primary goal of SKOMOBO is to provide a low-cost sensor platform, we first designed a circuit board (also known as breakout board) where we could mount a number of different air quality sensors. The most important aspect of designing our own circuit board in-house by our engineering team was the flexibility and customizability. It gave us the chance to test different sensors and compare the price competitiveness without sacrificing quality. It also gave us the opportunity to check the compatibility between different monitoring sensors and to choose the combination among sensors that work best.

Mounting several sensors on a single integrated circuit board has advantages compared to multiple and separate sensor board designs such as energy use and cost. By placing different types of air quality sensors within a single circuit board, it enables us to monitor and compare different aspects of air pollutant sources. Figure 1 illustrates the printed circuit board design.

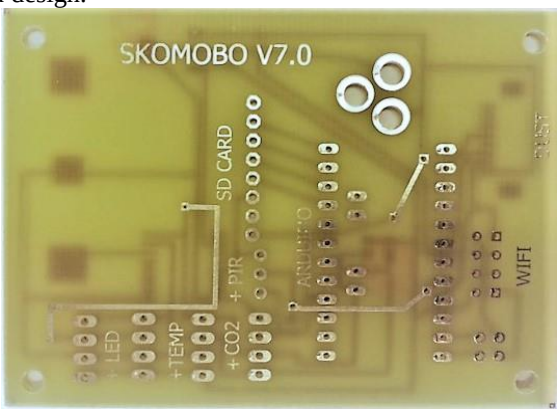


Figure 1: Blueprint for SKOMOBO circuit board

In the center of the circuit board, we have an Arduino Pro Mini [14] which runs at 5 voltage and 16 MHz processing power as a microcontroller board. It has 14 digital input/output

pins (of which 6 can be used as Pulse Width Modulation (PWM) outputs for controlling the analogue circuits with microprocessor's digital outputs), 6 analog inputs, and on-board resonator, a reset button, and holes for mounting pin headers. The six pin header can be connected to an FTDI (Future Technology Devices International Ltd) breakout board to provide USB power and communication to the board.

B. Sensors & Auxiliaries

The circuit board currently has the pins to connect four different air quality related sensors: a temperature/RH sensor, a CO_2 sensor, a particular matter (PM) sensor, and a PIR sensor. These sensors have been carefully chosen to capture different aspects of air qualities with the focus in classroom environments.

Figure 2 illustrates the breakout board with all sensors and auxiliary modules plugged into.

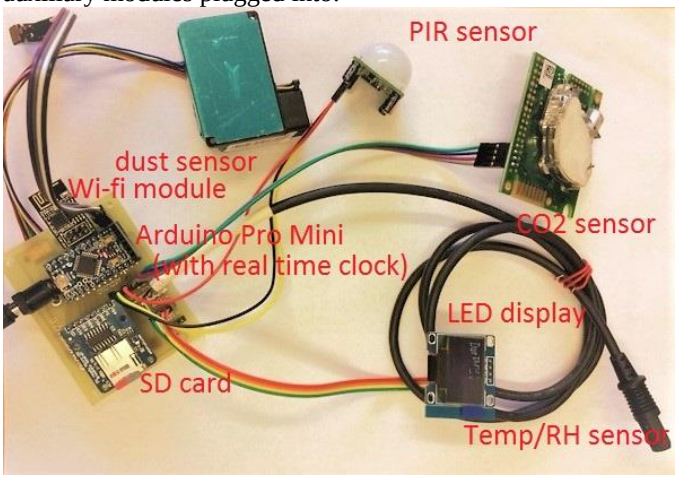


Figure 2: SKOMOBO with sensors & auxiliaries (enclosure removed)

A *temperature/relative humidity (RH) sensor* is used to capture the combination of classroom temperature and relative humidity. After evaluating number of sensors, our choice of the sensor was Telaire T9602 Temperature and Humidity sensor [15] which had the proven record of a reliable measurement. The main concern to plug this sensor to our circuit board was deciding the location of the sensor. As this sensor is sensitive to heat, it was important that the sensor was placed away from the main microcontroller and from other sensors to avoid the heat radiated from them which could interfere with the accuracy of data reading.

Carbon dioxide (CO_2) is often used as a proxy to estimate the stuffiness of the classroom. The *CO_2 sensor* is built to capture the carbon dioxide level. The use of this particular sensor is expected to show the ventilation level in classroom and whether there is sufficient fresh air flowing in to the classroom for students. We use SenseAir K30 model [16] which belongs to a group of Non-dispersive infrared (NDIR) CO_2 sensors to carry on the task.

A *particular matter (PM) measurement sensor*, also informally know as a dust sensor, is used to collect the level of dust densities in the classroom. The dust sensor is expected to show the quality of air the students breathe in. We choose PMS3003 dust sensor from Plantower for our project after discovering a reliable lab rest result [17].

A *passive infrared sensor (PIR) sensor* is an electronic sensor which measures infrared (IR) light radiating from objects in its field of view. The sensor is often used as a motion detector. We use PIR sensors (with the model number Duinotech XC-4444 [18]) to capture whether students present in a classroom and to investigate the relation between the student occupancy and the effects of CO_2 and dust emission.

Our circuit board also provides a number of pins to support other auxiliary modules that are important to our project. These include: a SD card, a Wi-Fi module, a real time clock and a LED screen.

We added a *SD card module* in our breakout board to store data collected from all our air quality measuring sensors. We use a SD card as a primary storage facility. The write operation to store the data is performed every minute which evaluated to work well with the given Arduino memory size and processing power.

We added a *Wi-Fi module* as well to transfer the sensor data to our secondary storage server to ensure data safety and reliability. The secondary storage server is located remotely away from the schools. For data transmission, we also deploy a Wi-Fi adapter (also commonly known as a Wi-Fi hotspot) at each school. The Wi-Fi module gets Internet connection via a Wi-Fi hotspot then transmits the sensor data after it is written in the SD card.

By default, the Arduino microcontroller do not support any means to find out time when it was important to us that we also record the time when the sensor data were collected. To solve the issue, a *real time clock* was added to the breakout board. Each time sensor data were sent, they were logged into the SD card along with the time. This capability is expected to be useful for time-series based data analysis which is to be performed at a later stage.

A *LED screen* is also added in our breakout board to display sensor data or error messages.

C. Voltage Mismatch

One of the significant engineering challenges we faced in the early stage of the hardware development was voltage conversion issue. Unfortunately, there was voltage mismatch between the Arduino microcontroller which runs on 5V and the sensors send signals on 3.3V. We had 3 different devices, Temperature/RH, real time clock and Wi-Fi module that runs on 3.3V and could get damaged if connected directly to the 5V Arduino.

In the first version, a logic level converter was used to safely step down 5V signals to 3.3V and step up 3.3V to 5V at the same time for the two bi-directional I2C data lines. This was necessary as we had I2C devices that work only on 3.3V IO (see Figure 3 for devices using different protocols and their relative voltage supplies). Additional pull up resistors the I2C IO lines were connected to 5V. We also made a version without a logic level converter and connected the pull up resistors to 3.3C which worked well and is implemented in our latest version.

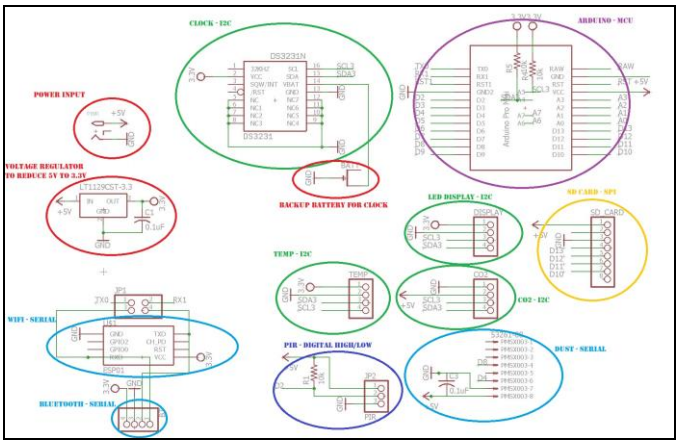


Figure 3: sensors with different protocols and voltage supplies

The Wi-Fi works through serial which works like conventional signal by changing the voltage levels. As the Wi-Fi works on 3.3V and not 5V, we needed to step down the Tx line from Arduino to 3.3V. A simpler way was to use a voltage divider. But we found out later that it does not support high baud rates. After further research, we found a Wi-Fi module which has a 5V tolerant capability so it can directly be connected to the Arduino without using a voltage divider.

IV. Software Engineering

A. *Software-based System overview*

In this section, we provide the details of the system overview illustrating how different components in the system interact between software we use in our project (overview illustrated in Figure 4).

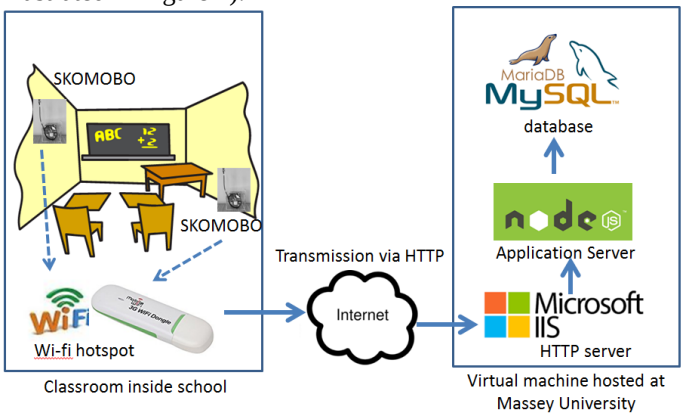


Figure 4: SKOMOBO overall system overview

At each school, a number of SKOMOBO boxes are deployed across multiple classrooms. Strategically deployed SKOMOBO boxes collect different air-quality related sensor data using their various sensors. A Wi-Fi hotspot is placed to receive Wi-Fi transmission of sensor data, as a form of HTTP request header, produced by each SKOMOBO box. The HTTP request header is routed over the Internet by the Wi-Fi hotspot using a 3G connection.

On a virtual machine running at Massey University, it runs a number of servers to process messages. A Microsoft IIS server installed in the virtual machine, which acts as a HTTP server, listens to any incoming HTTP requests. Once a HTTP request arrives, the HTTP server forwards the HTTP payload to Node.js application server. Node.js examines the HTTP payload and processes the sensor data which was contained in the HTTP header. The sensor data is read, formatted, and then logged onto a MariaDB database.

B. *Data Acquisition*

One of the most challenging tasks in developing a sensor platform is to deal with firmware(s) that interact with sensors which often use different underlying protocols. These protocols use different packet format and carry different data type (e.g., from row voltage signals to bits). Such heterogeneous nature of sensors makes it difficult to read and interpret data.

I2C devices: The CO_2 sensor, temperature/RH sensor and the real time clock use the I2C protocol. The I2C protocol uses two wires for transmitting data, SDA for transmitting the messages and SCL for managing a serial clock so that both participating devices do not have to agree on a speed. We decided to use this protocol for these sensors because they are all located on the same circuit. This allows multiple devices to share the same connection easily. These sensors are based on digital signals so it is also more accurate and reliable than an analogy system over short distances.

The I2C protocol runs between two devices, the one act as a master and the other act as a slave. One master can have many slaves that each has a unique address to identify it within the network. In our system, the Arduino is the master and the sensors are the slaves as seen in Figure 5.

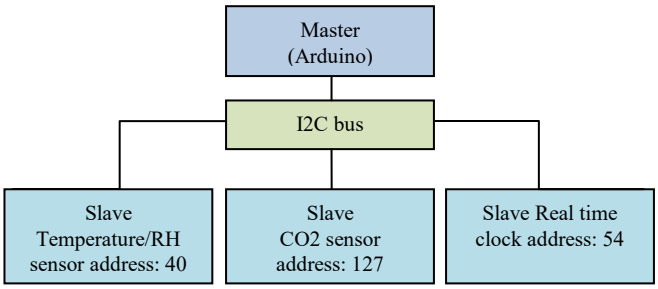


Figure 5: I2C protocol between master and slaves

For the Arduino to get data from the sensors, the addresses of the sensors had to be found. The only way to found out the addresses with no documentation available was by running test code that tried every possible address from zero to 127 and then logs whether or not a device is found at that address. Once addresses are found, two devices can communicate. In I2C protocol, only raw byte packets are communicated. It is usually left to the developers to find multiple values stored in different regions then process the data using different equations. For example, we use the following equation to calculate humidity after extracting the bytes from different regions in the I2C packet. Note that `rH_High [5:0]` indicates the extraction of 5 bits from the high region of the packet while `rH_Low [7:0]` indicates the 7 bits from the low region of the packet.

```
humidity = (rH_High [5:0] x 256 + rH_Low
           [7:0]) / 16384 x 100
```

Serial Devices: The dust sensor and the Wi-Fi module communicate with the Arduino using the serial protocol. This protocol only allows two devices to share the same connection and also uses two wires typically referred to as TX for transmitting and RX for receiving. To transmit data both devices must agree on a transmission speed so that the packets can be separated at a regular time interval and interpreted correctly. There are no checks built into the protocol that allow the devices to work out if there is a speed mismatch, that has to be determined by the developer of the application. For example, for our Wi-Fi module, we created a direct connection between a development machine and the Wi-Fi module using a USB to serial cable. Then we used a serial monitor application to send messages to the Wi-Fi module at different speeds. The testing ceased once a legible OK message was received from the module, indicating that it understood the command.

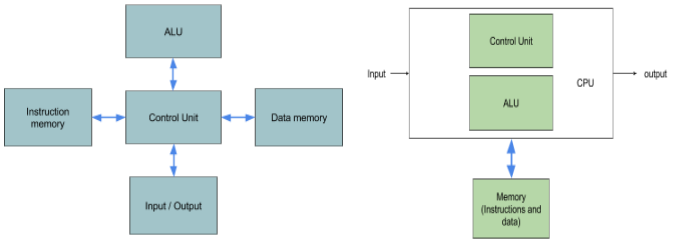
Digital Devices: Only the PIR sensor in our circuit makes use of raw digital signals consisting of high voltage levels, representing binary 1s and low voltage signals, representing binary 0s. Digital devices typically only send data on one line that has an electric signal passed through it based on certain conditions. For example, the PIR sensor in our system only sends an electric signal when it detects motion in front of it. Working with digital signals is simpler than other protocols as interpretation of digital signals is straightforward with no calibration or calculation. For example, when motion is detected, the PIR sensor sends a value "1" which we can interpret as "motion detected" and treat "0" as "no motion detected".

C. *Data Logging*

During start up, a CSV file is created on the SD card and comma separated column headings are stored inside it. All the individual values for each sensor are then passed to the snprintf_P function that is included in the Arduino library. This function then stores all the data separated by commas ending in a new line in a character array buffer of a fixed size. The buffer is then written character by character to the SD card by transmitting the data using the SPI protocol, which functions like I2C however slaves are selected by sending a signal down a separate wire called the select line, where it then stores all the data in the CSV file. The most common issue that arises with this method is the character array buffer overflowing, resulting in the message becoming truncated, due to certain values being much higher than expected.

D. *Memory Size Constraints*

Arduino Pro Minis are manufactured using a variety of different Atmel chips that have different hertz rates which is essentially the processing speed. The model used in this project has a hertz rate of 16 MHZ. Compared to modern computers that have a processor with a hertz rate that is in the GHz ranges, such as intel i7 processors, it is very slow. To make up for this the Atmel manufacturers had to utilize efficient memory retrieval.



(a)Harvard Memory Architecture (b)Von Neumann Memory Architecture

Figure 6: different memory architecture

It achieves this by using the Harvard memory architecture which separates the programs storage space for all the instructions from the main data of the application. It also has two separate busses one for fetching data and one for fetching instructions allowing it to perform both actions simultaneously instead of only being able to do one or the other like in a von Neumann architecture. The comparison between these two memory architecture is illustrated in Figure 6.

The chip allocates three different memory spaces and store different types of both instruction and data.

- 32 Kilobytes of Flash memory (also known as programming memory) two of which are used up by the Arduino's bootloader
- 2 kilobytes of SRAM which is used as a space for storing variables
- 1 kilobyte of EEPROM which provides a permanent form of storage that could allow for some degree of crash recovery, with a limited number of writes, which our project does not utilise.

As soon we found out, the memory management and underlying memory architecture were an important issue to understand to avoid many undesirable effects. For instance, originally our system used local strings extensively, as well as function arguments and next to no global variables to avoid issues with overwriting data that is used in other regions of the codebase due to mutability. This however resulted in a large amount of code duplication especially with regards to message formatting for storage on the SD card, displaying on the LED display and sending it as HTTP GET requests to the server, all of which require unique styling demanding heavy memory use and allocation.

We started realising the impact of memory size when we integrated a Wi-Fi module. The library that we used for the Wi-Fi module used up a extra 20% of memory and we started observing instability in the data reading.

Another issue that formed from this was heap fragmentation. Heap fragmentation is a phenomenon caused by variables stored in the heap changing size. Since the variables do not have limits placed on them the firmware simply searches through the heap for a free memory block where it can fit the newly sized data and then slots it in. When the data inevitably changes size, or gets deleted it leaves behind a gap

due to other variables being added and removed whilst current ones are stored.

There is no order to the placement of the memory besides this basic rule which results in tiny gaps forming in the memory space that nothing can fit in. These gaps then cause the upper limit of the heap to continue growing until it reaches the stack and eventually starts overwriting values stored in its memory addresses. This then caused symptoms such as unstable wireless connectivity due to the main Wi-Fi commands (i.e. AT commands) being malformed.

To cut down on the memory usage code changes had to be made in a number of places:

- First, we used the Flash macro. This allowed us to reduce our SRAM memory usage by about 20% by allowing us to store most of our static information in Flash memory.
- In our latest architecture for our application all function arguments were replaced with singular primitive global variables that store data for each individual sensor and a single global character array buffer that can currently store 100 characters for the formatted messages.
- All instances of the String library were also replaced with their C++ standard character array equivalents. For instance, string concatenation was replaced with the sprintf function calls. In addition the _P variants of these functions were used which means that the formatting gets stored in Flash memory. The n modifier was also used for each of these functions allowing for an extra argument specifying the length of the final array. This prevents dynamic allocation issues by truncating the message after it goes past the allocated space.
- Another technique that was used to reduce memory usage is that single characters get read out of flash and then printed to the LED display. This means that the entire message can be printed whilst only ever using one byte of SRAM to store the character.

These changes resulted in another 20% memory reduction. In addition it allowed the compiler to give warnings if memory usage is becoming too large because it can calculate the size of all the variables. It also means that heap fragmentation was no longer an issue.

E. Networking Constraints

The ESP8266 [19] was chosen as a Wi-Fi module for wireless transmission of the sensor data. The model was chosen because it is easy to configure, connects directly to a hotspot (also working as a router) with a strong signal, and runs a full TCP/IP stack on its chip.

One major issue with the Wi-Fi module was its severe limitation with throughput rate. This combined with a restriction on the number of characters our Wi-Fi module can transmit and the slow transmission rate of the hotspot we were using resulted in significant amounts of packet loss.

Compared to other costly Wi-Fi modules which can send data up to approximately 4.5 Mbps, our module maxed out at about 7 Kbps which was very slow. To enhance the performance, we could send our data in UDP mode which guarantees fast transmission rate at the cost of reliability. As a reliable delivery of data transmission between the sensors and the sever was necessary, we decided to keep sending our data via HTTP though it was slow and to find other solutions.

Based on our lab test, our 3G hotspot worked significantly slower both download and upload speeds compared to other types of Wi-Fi hotspots (see Figure 7).

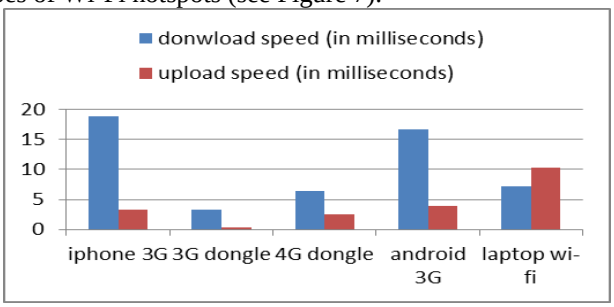


Figure 7: hotspot data speed comparisons

To keep production costs low, changing sensors and modules were not a desirable choice since the throughput rate could be mostly fixed using software.

One software approach was to format packets as HTTP GET requests as it allowed for simpler HTTP packets to be sent that do not need to utilize chunking or specifying an encoding type which all require extra processing. We simply encoded the data inside the URL using delimiters to save on the number of characters used thus reducing the payload size.

The second approach was to increase timeouts. The client that sends data, in this case the Arduino typically sets a timer to ensure a successful data transmission within a reasonable timeframe. If it does not receive an acknowledgement from the receiver (i.e., the Massey server), it timeouts and tries to retransmit. However, due to the low throughput of both the Wi-Fi module and the hotspot, it took longer to receive data and acknowledge. Different timeout intervals were tested until it was found that the interval had to be set to about five minutes for any noticeable difference. However, the timeout was eventually reduced to one minute so that it does not block the rest of the systems operations for that period of time.

V. Experimental Results

In this section, we illustrate the overall cost involved in producing SKOMOBO and how SKOMOBO performs in a field study. Our field study demonstrates those low sensors that are used in our project produces accurate and reliable data.

A. Cost Evaluation

The cost breakdown of different sensors and components we use in SKOMOBO is shown in Table 1. As can be seen, the SKOMOBO provides a significant cost reduction compared to what is commercially available today. Considering that some of IAQ monitoring tools available today in the market costs from $2,500USD to $4,000USD, the SKOMOBO promises up to 94% cost reduction. This illustrates that SKOMOBO can be a viable solution to deploy and learn about air quality in as many classrooms due to its low cost. Note that the total do not

contain the cost involved in an enclosure, a Wi-Fi hotspot or data plan cost.

Table 1 : price of sensors and other components

Sensors & auxiliries	Model	Price (USD)
Temperature/RH sensor	Telaire T9602-3-D-1	38.56
CO_2 Sensor	SenseAir K30	91.5
Particular Matter (PM) sensor	Plantower PMS3003	15.34
PIR sensor	Duinotech XC-4444	5.12
Microcontroller	Arduino Pro Mini	9.95
Cricurt Board	Seeed Studio	20.13
MicroSD card breakout board	Adafruit	7.50
Micro SD card	SanDisk 8GB	9.52
WiFi module	SparkFun-ESP8266	6.95
Power adapter	PowerTech Plus MP-3144 5VDC, 1.0A	9.52
Total		**214.09***

*This price does not contain time cost in building SKOMOBO.

B. Performance Evaluation

We conducted a field study to evaluate the feasibility of SKOMOBO. Three SKOMOBO boxes were placed in 2 classrooms in a nearby primary school from Massey University and the data reading from the three SKOMOBO boxes were monitored. The study was done over 2 weeks in the early August 2017. Following graphs show typical data reading from a specific day to illustrate the details and interpretations from the sensors. Note that the data reading from other days were consistent and showed similar patterns. All data were recorded at 1-minute interval.

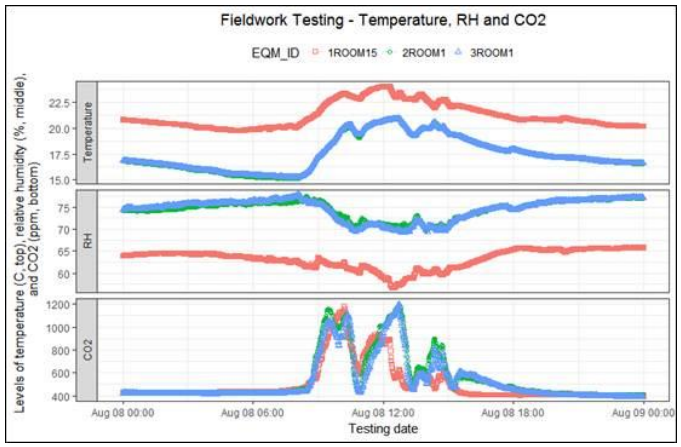


Figure 8: Data reading from SKOMOBO boxes on temperature, relative humidity (RH), and CO_2

Figure 8 illustrates the data measurements on temperature, relative humidity and CO_2 among 3 SKOMOBO boxes. As seen from the graphs, all three data reading from SKOMOBO responded compatibly to school activities;

- For example, from the temperature graph (the top graph) shows the temperature slowly rising from about 7am. The temperature was the warmest during about 8am when the school started and students entered into the classroom till about 3pm the time when the school finished and the majority students left from their classrooms. We witness a slight temperature fluctuation between these hours as some students left and came back again during various break times.
- Relative humidity (the middle graph) illustrates constant change relative to temperature. As can be seen, the result of rising temperature corresponded well with relative humidity falls and vice versa. The rise and fall between the temperature and relative humidity was in a strong agreement accurately reflecting the relationship between temperature and humidity.
- CO_2 (the bottom graph) results were compatible as well. The level of CO_2 sharply raised in the school hours between 8am and 3pm - the time when the majority of the students present in the classroom. The reading from all three SKOMOBO boxes was in the consistent pattern showing the highest level of CO_2 when the classrooms were occupied with the students.

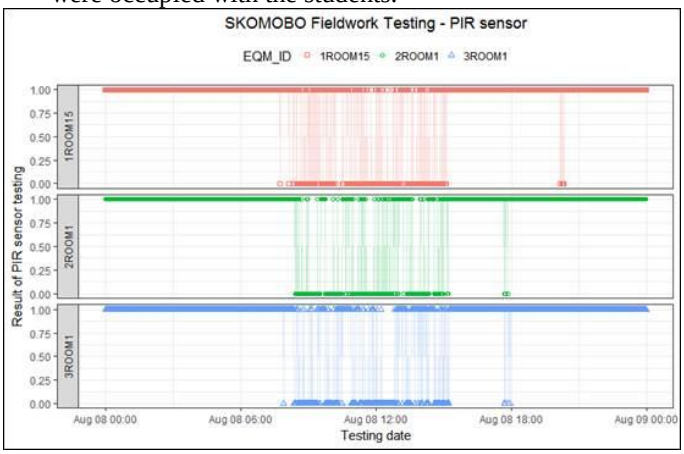


Figure 9: Data reading from SKOMOBO boxes on PIR sensor

Figure 9 illustrates the data reading from PIR sensor. As can be seen from the graph, all three sensors in the different classrooms correctly detected the motion (i.e., student's presence in the classroom) between school hours.

In overall, the results of data reading showed an excellent indication of accurate measure from all three different SKOMOBO boxes placed in different classrooms. Though the discrepancies exists between our reading and some recommended reading, our experimental analysis shows that the gap is compatible (i.e., within the acceptable range) and is expected to give the information that would allow us to understand the bigger picture in indoor climate and ventilation practices in New Zealand schools.

VI. Conclusion

Recent developments in low-cost sensor R&D combined with remote logging and data transmission technologies have opened up new opportunities for producing low-cost Indoor Air Quality (IAQ) monitoring system. This enables opportunities to study indoor climate in a large number of New Zealand classrooms and ensure that children are learning in healthy environment.

Existing studies in the area however often only focus on developing new technologies, system architecture and networking performance rather than describing practical guidelines for many impending hardware and software engineering issues. In this paper, we developed a low-cost IAQ monitoring platform called SKOMOBO which utilizes low-cost sensors to bring down the cost of the equipment so that it is affordable for many schools. Through extensive experiments and evaluation, we were able to understand the characteristics and limitations of developing a low-cost sensor platform and their practical implications for deploying them in school classrooms.

We plan to deploy SKOMOBO in 30 schools in 90 classrooms across New Zealand to understand air quality implication and ventilation practices in different schools. We will continue to apply refinement to improve our system.

In addition to showing the feasibility of monitoring indoor environment for a fraction of the usual cost, we also envisage that our project to be used as a part of a STEAM (Science, Technology, Engineering, Arts and Mathematics) learning educational tool. In collaboration with high school students, we have a firm plan to develop an app that would allow children to view data from SKOMOBO in their classroom. The app will facilitate student engagement with their indoor environment. They will be able to link to SKOMOBO and retrieve data to use in the math, science and ICT curriculum to learn about the health values of their classroom.

ACKNOWLEDGMENT

The authors would like to thank Building Research Association of New Zealand (BRANZ) for the financial support with the project reference number LR0513.

REFERENCES

[1] M. Mendell, E. Eliseeva, M. Davies, M. Spears, A. Lobscheid, W. Fisk and M. Apte, "Association of classroom ventilation with reduced illness absence: a prospective study in California elementary schools", *Indoor Air*, vol. 23, no. 6, pp. 515-528, 2013.

[2] M. Turunen, O. Toyinbo, T. Putus, A. Nevalainen, R. Shaughnessy and U. Haverinen-Shaughnessy, "Indoor environmental quality in school buildings, and the health and wellbeing of students", *International Journal of Hygiene and Environmental Health*, vol. 217, no. 7, pp. 733-739, 2014.

[3] J. Gao, P. Wargocki and Y. Wang, "Ventilation system type, classroom environmental quality and pupils' perceptions and symptoms", *Building and Environment*, vol. 75, pp. 46-57, 2014.

[4] Y. Jiang, L. Shang, K. Li, L. Tian, R. Piedrahita, X. Yun, O. Mansata, Q. Lv, R. P. Dick, and M. Hannigan, "MAQS: a personalized mobile sensing system for indoor air quality monitoring." *In Proceedings of the 13th international conference on Ubiquitous computing*, pp. 271-280. *ACM*, 2011.

[5] A. Caron, B. Hanoune, N. Redon, and P. Coddeville. "Gas sensor networks: relevant tools for real-time indoor air quality indicators in low energy buildings." *Proceedings of the Healthy Buildings Europe 2015 Conference*. 2015.

[6] J. Kim, C. Chu and S. Shin, "ISSAQ: An Integrated Sensing Systems for Real-Time Indoor Air Quality Monitoring", *IEEE Sensors Journal*, vol. 14, no. 12, pp. 4230-4244, 2014.

[7] P. Kumar, A. Skouloudis, M. Bell, M. Viana, M. Carotta, G. Biskos and L. Morawska, "Real-time sensors for indoor air monitoring and challenges ahead in deploying them to urban buildings", *Science of The Total Environment*, vol. 560-561, pp. 150-159, 2016.

[8] Y. Xiang, R. Piedrahita, R. P. Dick, M. Hannigan, Q. Lv and L. Shang, "A Hybrid Sensor System for Indoor Air Quality Monitoring," *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, Cambridge, MA, 2013, pp. 96-104.

[9] G. M. Revel, M. Arnesano, F. Pietroni, J. Frick, M. Reichert, K. Schmitt, J. Huber, M. Ebermann, U. Battista, and F. Alessi. "Cost effective technologies to control indoor air quality and comport in energy efficient building retrofitting" *Environmental Engineering & Management Journal (EEMJ)* 14, no. 7 (2015).

[10] S. Abraham, and X. Li. "A cost-effective wireless sensor network system for indoor air quality monitoring applications." *Procedia Computer Science* 34 (2014): 165-171.

[11] A. Zanella, N. Bui, A. Castellani, L. Vangelista and M. Zorzi, "Internet of Things for Smart Cities", *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22-32, 2014.

[12] M. Penza, D. Suriano, M. G. Villani, L. Spinelle and M. Gerboles, "Towards air quality indices in smart cities by calibrated low-cost sensors applied to networks," *IEEE SENSORS 2014 Proceedings*, Valencia, 2014, pp. 2012-2017.

[13] R. Honicky, E. A. Brewer, E. Paulos, and R. White. N-smarts: networked suite of mobile atmospheric real-time sensors. *In Proceedings of the second ACM SIGCOMM workshop on Networked systems for developing regions (NSDR '08).* ACM, , pp. 25-30, 2008.

[14] "Arduino Pro Mini", Store.arduino.cc, 2017. [Online]. Available: https://store.arduino.cc/usa/arduino-pro-mini. [Accessed: 13- Aug- 2017].

[15] [Online]. Available: http://www.mouser.com/ds/2/18/AAS-920-638F-Telaire-T9602-060316-web-850549.pdf. [Accessed: 13- Aug- 2017].

[16] [Online]. Available: http://www.senseair.asia/Datablad/K30_Platform_description.pdf. [Accessed: 13- Aug- 2017].

[17] "The Plantower PMS3003 Air Quality Sensor experiment", aqicn.org, 2017. [Online]. Available: http://aqicn.org/sensor/pms3003/. [Accessed: 13- Aug- 2017].

[18] [Online]. Available: https://www.jaycar.com.au/arduino-compatible-pir-motion-detector-module/p/XC4444. [Accessed: 13- Aug- 2017].

[19] "ESP8266 WiFi Module - Espruino", Espruino.com, 2017. [Online]. Available: https://www.espruino.com/ESP8266. [Accessed: 13- Aug- 2017].