Digital Communications and Networks 6 (2020) 23–30

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

Real-time air pollution monitoring with sensors on city bus

Sami Kaivonen, Edith C.-H. Ngai *

Department of Information Technology, Uppsala University, Sweden

ARTICLE INFO

Keywords:
Smart city
Internet-of-Things
Mobile sensor network
Air pollution monitoring

ABSTRACT

This paper presents an experimental study on real-time air pollution monitoring using wireless sensors on public transport vehicles. The study is part of the GreenIoT project in Sweden, which utilizes Internet-of-Things to measure air pollution level in the city center of Uppsala. Through deploying low-cost wireless sensors, it is possible to obtain more fine-grained and real-time air pollution levels at different locations. The sensors on public transport vehicles complement the readings from stationary sensors and the only ground level monitoring station in Uppsala. The paper describes the deployment of wireless sensors on Uppsala buses and the integration of the mobile sensor network with the GreenIoT testbed. Extensive experiments have been conducted to evaluate the communication quality and data quality of the system.

1. Introduction

Internet-of-Things (IoT) is a network of devices that can collect and exchange data. Connected devices may have embedded electronics, sensors and network connectivity which enable these devices to connect and collaborate with other IoT devices [1]. The IoT has been used in automobiles and city infrastructures, such as smart lighting, water, power, and cooling and alarm systems [2–5].

Air quality monitoring is a subject for many research projects and community-based initiatives. Air quality has become a major concern for many cities around the world. Poor air quality in urban areas may cause various health problems for people who are exposed to it in their everyday life [6]. The main causes for poor air quality are vehicle exhausts and industrial sites located close to urban areas. Many cities have deployed a small number of expensive monitoring stations for monitoring air quality. However, the deployment of city infrastructures costs a lot of time and money. Due to the high costs, there are often only a few number of stations to provide limited coverage in the city. Recently, research projects have been conducted to investigate low-cost sensors so that they can be deployed more widely in the city.

The work in this paper was conducted at Uppsala University in Sweden as part of a nationally funded project called the GreenIoT. The GreenIoT project has developed a testbed with wireless and stationary sensors for air quality monitoring in the city center of Uppsala [7]. In order to complement the coverage of the stationary sensors, we've planned to deploy air quality sensors also on public transport vehicles. Mobile sensors allow measurements to be taken at different locations in the city without the constraints and extra costs of installing sensors on city infrastructures. They are also able to increase the coverage of the sensing area significantly.

The first goal of this work is to identify a set of bus routes that can provide good coverage of the town and pass through the highly polluted areas that require attentions. Route planning is studied in this work to select bus routes that can acquire measurements from the important locations. It is conducted via image analysis on a bus route map provided by the local bus company, Upplands Lokaltrafik (UL). The second goal is to deploy the sensor on a public vehicle and evaluate the capability of a moving sensor in comparison with that of a stationary sensor. The sensor program developed has to be robust enough for long periods of operation without maintenance. The final goal of this work is to evaluate sensor performance and data quality. The evaluation of sensor performance is based on data collected by the sensors deployed on public vehicles over a long period of time. And the data quality is evaluated by comparing the measurements with those from other sensors deployed in the city of Uppsala.

2. Related work

A number of research projects have been conducted to measure air quality using the IoT or wireless sensor networks [8,9]. These projects include measurements of different types of pollutants in the air using stationary or mobile sensors. For example, University of Patras evaluated the power consumption of the Waspmote platform in 2015 [10]. Waspmote platform was developed by a company called Libelium, which

* Corresponding author.
E-mail address: edith.ngai@it.uu.se (E.C.-H. Ngai).

https://doi.org/10.1016/j.dcan.2019.03.003
Received 2 October 2018; Received in revised form 9 March 2019; Accepted 12 March 2019
Available online 16 March 2019

provides different types of sensors, radio technologies, open source SDK (Software Development Kit) and API (Application programming interface) for sensor network development. The paper focused on identifying critical operations and implementing a setup for power consumption measurements in wireless sensor network. Similarly, University of the Armed Forces developed a wireless monitoring system for air quality measurement [11]. The study also attempts to deliver a system with hardware, software and firmware solutions for measuring air quality. The system was developed on an Arduino platform using a network gateway to connect the sensor nodes to the Internet. It measured carbon monoxide (CO)and carbon dioxide (CO_2) concentrations in the city of Quito, Ecuador.

Besides stationary sensor networks, mobile sensors have also been investigated to support air pollution monitoring. OpenSense was a project conducted in Zurich, Switzerland [12]. In the OpenSense project, a tram was used as a moving platform for carrying the sensors. Communication between the sensors and the cloud was done using GPRS (General Packet Radio Services). In this paper, we consider deploying sensors on city buses, which provide wider coverage in the city and more options and flexibility in the selection of bus routes. City buses also have different installation infrastructures, more dynamic speeds, and stop and moving patterns.

Another project in which air quality was monitored with an autonomous wheeled rover was conducted in 2016 by Sapienza University of Rome [13]. The sensor measured methane (CH_4), ethylene (C_2H_4), ammonia (NH_3), benzene (C_7H_8), LPG (C_4H_{10}), CO_2, CO and nitrogen oxides (NOx). The rover was developed by Sapienza University with GPS sensors, three axis gyros, accelerometers and magnetometer, so that it was able to navigate and avoid obstacles on the path to scan the pollution status over a large area.

3. Background

3.1. The GreenIoT project

The GreenIoT project aims to research on an energy-efficient IoT platform for the general public to develop innovative applications based on open data provided by the GreenIoT testbed [7]. A case study for the GreenIoT is implemented in Uppsala, the fourth largest city in Sweden. It includes an IoT testbed deployed in the city center for monitoring air pollution. Uppsala occasionally exceeds the EU standards on the particulate levels, especially in winter and spring, which gives an objective for the project to reduce air pollution through active monitoring, traffic management and better city planning.

Fig. 1 shows the architecture design of the GreenIoT testbed [7]. Stationary sensors are connected to a sensor gateway via a low energy consumption network protocol called IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN). The sensor gateway relays sensor data to the GreenIoT cloud in Sensor Markup Language (SenML) format via WiFi connection. In the cloud, the data is stored in a NoSQL (not only Structured Query Language) database and published to a MQTT (Message Queuing Telemetry Transport) Broker. The data is stored into MongoDB (MongoDB is a cross-platform document-oriented database program)

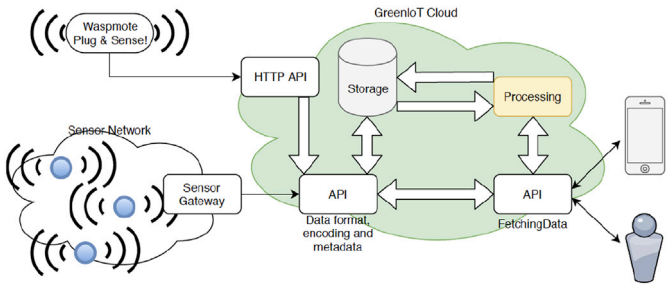

Fig. 1. The system architecture of the GreenIoT testbed.

collections using SenML format which is a regular JSON (JavaScript Object Notation) object with a pre-agreed structure [14]. The cloud also provides an HTTP (Hypertext Transfer Protocol) API for user applications to access the GreenIoT sensor data from the database [7]. In this project, we use the GreenIoT cloud for data storage, processing and visualization, and the sensors on city buses communicate with the GreenIoT cloud server through 4G connectivity and HTTP in the application layer.

3.2. Libelium Waspmote sensor

The sensor used on the city bus is the Libelium Waspmote Plug & Sense! model called Smart Environment PRO [15]. The device includes removable sensor probes for measuring nitrogen dioxide (NO_2), CO, temperature, humidity and pressure. It contains a rechargeable 6600 mAh lithium battery and chips with a solar panel and a USB cable, both of which can be used to recharge the device. When charging the device with a USB cable, a regular USB input voltage of 5 V is required. The Smart Environment PRO supports 4G data connection with GPS location tracking. It uses three antennas (the cellular main antenna, the cellular diversity antenna, and the GPS antenna) to provide these connections. The sensor model with 4G radio also includes a SIM-card slot and a micro-USB connection for programming the sensor.

3.3. Public transport vehicle

The sensor is deployed on a public transport vehicle owned by Gamla Uppsala Buss AB, a local bus company. The type of vehicle used in this project is the Lions City Hybrid bus from MAN Truck & Bus AG [16]. The Lions City Hybrid bus is powered by a diesel engine and an energy storage consisting of a set of Ultracapacitors, whose abbreviation is Ultracaps. Ultracaps are a new and alternative type of energy storage compared with a conventional battery system. Ultracaps are located at the front part of the vehicle on the roof inside a plastic compartment and are air-cooled by three fans at the back of the compartment. They are charged by a brake energy recovery system which utilizes the energy created by the brakes when the vehicle slows down and stores it into the energy storage. The diesel engine uses the stop and start system which enables the engine to shut down when it is not needed. A combination of these technologies enables the vehicle to leave the bus stops on electric power alone and therefore start moving exhaust free. A Lions City Hybrid bus saves 10000 L of diesel per year and reduces up to 26 tons of CO_2 output per year.

4. Methodology

In order to select optimal bus routes for the deployment of sensors, an image analysis algorithm is used to identify the route that cover the widest area in the city of Uppsala. Data from the sensors is then reported to the GreenIoT cloud via the mobile network through the HTTP interface. Data analysis is performed based on a variety of techniques to process data, which are implemented by Python scripts. Visualization of the sensor data is implemented via web programming on Google Maps to show the location and information of the measurements.

4.1. Route selection

The selection of bus route is done by a route coverage image analysis algorithm based on information received from the route coordinators in Gamla Uppsala Bus AB. The route coverage image analysis algorithm is developed based on a city bus route map from UL.com and analyzed using Matlab and Photoshop. The route map is a non-satellite image of Uppsala city. The algorithm estimates the percentage of area covered by different combinations of bus routes. There are 21 bus routes in Uppsala during the project time.

As shown in Fig. 2, the routes are marked in red on the map after processing. The image of the map includes 69×10^6 pixels and the bus

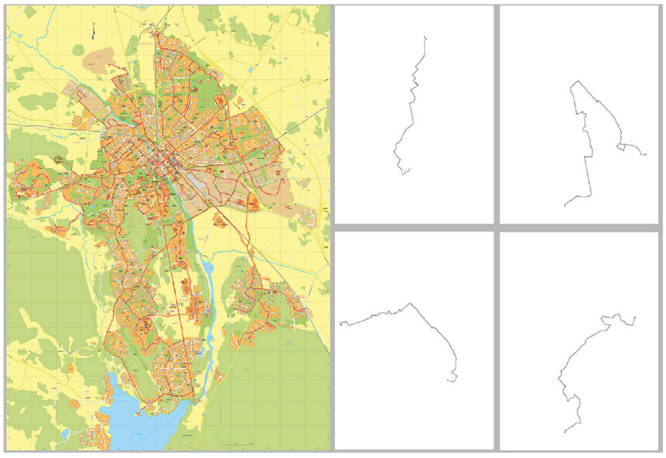

Fig. 2. The map that was used in the image analysis (left) and 50% reductions of four route extraction (right).

routes cover 1094584 pixels out of them. When calculating the coverage of a route in the city area, 1094584 pixels covered by the routes are considered as fully covered and serve as a base in comparison with any sub-area covered by a selection of routes.

The Matlab algorithm includes a few important parameters. The first one is used to set the number of routes that can be selected. The second one is optional and is for cropping a smaller area from the map image where the calculation is performed on. The third is optional and can be used to set areas that the route must pass through, forcing all the routes to pass through those mandatory areas.

4.2. Data collection

Fig. 3 shows the process of delivering data from the Waspmote Plug & Sense! sensor to the end user. The sensor composes a POST request from all measurements and sends it via 4G mobile network to the HTTP API in the GreenIoT cloud. The HTTP API then parses the values from the POST request and assembles a JSON object that is sent to the MQTT Broker. The MQTT Broker creates a topic for MQTT subscriptions of the data and adds new data to the Mongo database.

In order to get 4G mobile network connectivity, the sensor needed a SIM-card. Telia Sverige AB was chosen as a service provider and a SIM-card with 4G subscription was acquired from one of the project partners. The HTTP API was created with PHP so that it could easily process the incoming POST request with the measurements in its payload. The GreenIoT database is a NoSQL database called MongoDB in which the data is stored as JSON object. The stored JSON object follows an agreed data format, called Sensor Markup Language (SenML). SenML includes the name of the sensor responsible for the measurements, the time when the measurements are taken, and a nested JSON object that includes all the measurements.

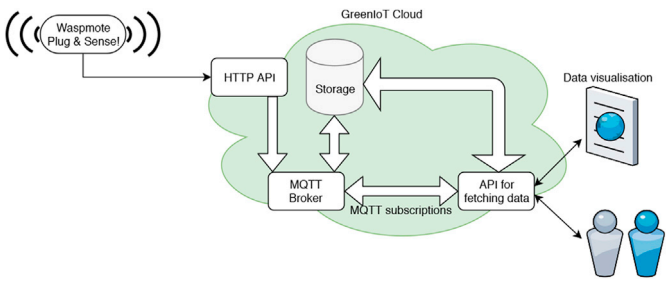

Fig. 3. Network diagram for data to reach the end users from the sensor.

4.3. Visualization

The data collected by the sensor on bus is displayed on Google maps using two different methods. The first method is to display the data with a heatmap and the second is to display the ten latest measurements on a map that is updated in real-time based on the location and movement of the bus.

Fig. 4 shows the heatmap layer on Google maps created with the data collected by the sensor on bus. The heatmap does not represent the air quality value but indicates which areas have the most data points. Every data point creates a circle that has a higher value in the middle and a lower value (lighter in color) on the outer skirts of the circle. Having multiple data points on a single location makes the circles overlap with each other and gives that location a higher value. Areas that have more data points close to each other appear darker on the map as the color changes to red after exceeding a specific threshold. By using a heatmap, it is easy to understand which areas of Uppsala are most effectively covered by the sensor on bus.

Fig. 5 shows the ten latest measurements taken by the bus sensor on Google maps. The map displays the ten most recently collected data points with clickable color coded markers and shows the measurement information in an information window which shows up when the marker is clicked. The information window contains the measurement time, the node-ID of the sensor, NO_2, CO and temperature, humidity and pressure values for that data point.

5. Implementation

The implementation of the project focuses on the deployment of the sensor on bus, the effort to prevent weather conditions from affecting the sensor measurements, and how the actual program of the sensor was designed to work. The sensor had limited mounting options on the roof of the bus where it could be installed and it had to function reliably in a long period. Implementation challenges were to protect the sensor from the sun and the rain to prolong its effective lifetime and to avoid any kind of technical malfunctions. At the time of the deployment, Uppsala had a late winter snowstorm, which made it necessary to cover up the sensor probes to prevent them from sinking into the snow and getting wet while still enabling them to take air quality measurements. By adding extra protection, the fluent air flow was disturbed, which became another challenge.

5.1. Deployment

The deployment of the first sensor was completed on 3 March 2018. It consisted of two phases: testing and actual deployment. The test was conducted in the central area of Uppsala with continuous access to the

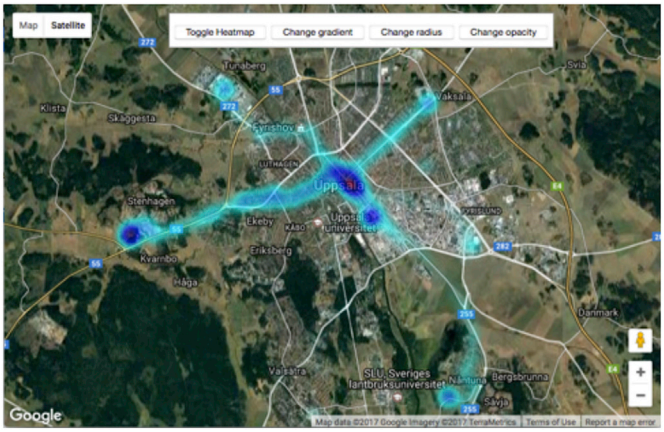

Fig. 4. Heatmap layer on Google maps composed of the data collected by the sensor on bus. The intensity of the color indicates the number of data points in the area.

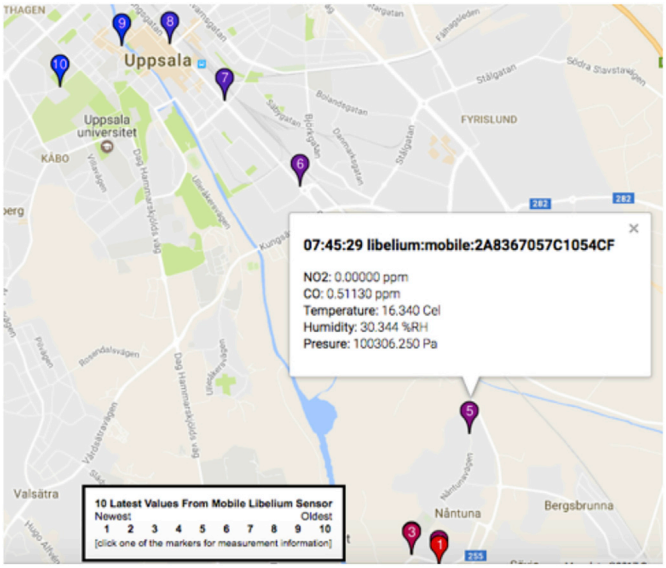

Fig. 5. Ten most recent measurements displayed on Google Maps.

sensor debug screen via a laptop and a USB cable. After successfully taking measurements and sending them to the GreenIoT database, the test was completed and the deployment was done in cooperation with Gamla Uppsala Bus AB. Before deploying the sensor on bus, it was necessary to plan multiple aspects of the deployment.

The hybrid bus offered continuous power supply for the sensor in 24–28 V, but the sensor could only take in 5 V via its USB port. Therefore, an adjustable step-down voltage converter, as shown in Fig. 6, was installed, which enabled a conversion from higher voltage to 5 V to fit the sensor's USB connection requirements. The adjustable step-down voltage converter was barely a chipboard without protection from environmental effects. To protect the voltage converter from bad weather conditions, a waterproof box was bought. The box also offered rubber seals on each of the two cable exits. Finally, the voltage converter was placed inside the battery compartment of the bus, which is shielded by the exterior walls and the roof.

The sensor was installed behind the battery compartment on the roof of the bus on the left side. To prevent the sensor from direct exposure to the sun, bad weather conditions and tree branches, a cover was made. Together with Gamla Uppsala Bus AB, we designed a cover in a way that it would neither affect the 4G signal too much nor block the sensors from taking readings while still providing a cover for protecting the sensor probes. The shield shown in Fig. 7 was made of metal to give the sensor a maximum cover for a long period of time.

5.2. Sensor program

The sensor code needs to be reliable and robust without requiring frequent maintenance and checkups. The basic code was developed

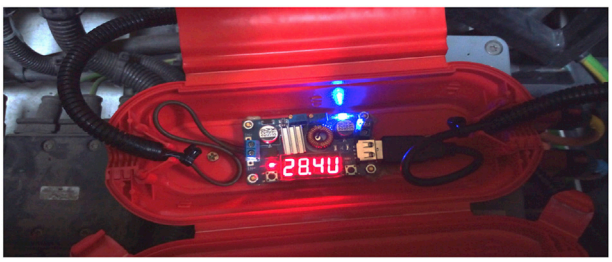

Fig. 6. The adjustable step-down voltage converter and the waterproof protective box.

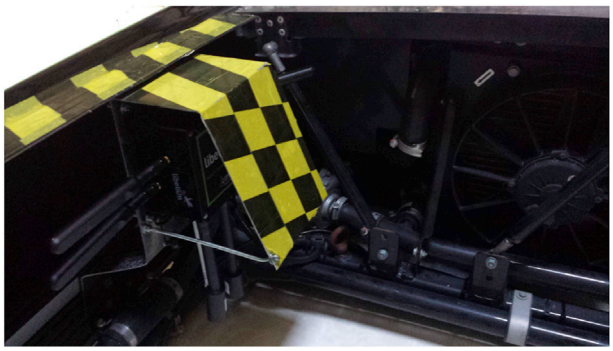

Fig. 7. Installation of the sensor on the bus.

based on example codes provided by the Waspmote IDE and added with more advanced functions. Every Waspmote Plug & Sense! program includes two functions: *setup* and *loop*.

The *setup* function executes only once after switching on the power or restarting the sensor. Setup formats the SD card of the device, initializes the Real Time Clock (RTC) time, and sets the node ID for the sensor. The RTC time is used to send the difference between the sensor starting time and the time when the measurement was taken to the HTTP API. The node ID is used as an identification name for the sensor when sending the measurements to the GreenIoT database. It is composed of two parts: a static string that is the same for every sensor that has the project code, and a unique serial code string that is different for every sensor.

The *loop* function is executed after the setup function is initiated and keeps executing as a loop till the power of the sensor is turned off. In the loop function, the *CO* and NO_2 sensor functions are firstly turned on for warming them up and the rest of the sensor functions are shut down by setting the sensor into the deep sleep mode. RTC is used to wake up the sensor from the deep sleep mode after 2 mins. When the sensor wakes up from the deep sleep mode, the 4G module is turned on and the GPS coordinates are fetched. In the Waspmote Plug & Sense! Smart Environment model, the GPS module is included in the 4G module. Therefore, the 4G module is turned on so that it is able to fetch GPS coordinates.

After the GPS coordinates have been received, the sensor takes measurements for the NO_2, *CO* and temperature, humidity and pressure. After all the data is acquired, the sensor stores the data into a PHP request string and tries to send the data to the GreenIoT cloud. If the 4G data connection fails to be established to the GreenIoT cloud, the PHP request is stored into a text file on the device's SD card. Once the 4G data connection is successfully established, the latest measurement and the ten most recent measurements will be sent to the cloud from the storage text file.

5.3. HTTP API

The Waspmote Plug & Sense! sensor data was planned to be published to the GreenIoT MQTT Broker, which in turn would enable the data for client subscriptions and add the data to the GreenIoT database. The sensor communicated through 4G cellular data connection and could send POST requests via that connection, therefore an HTTP API was made for enabling the sensor to publish data to the MQTT Broker. The API receives ten string variables in the POST request payload: Node ID, time from the moment the measurement was taken till the moment it reached the GreenIoT cloud, GPS latitude and longitude, the measurements from the sensors, and a debug value between one and zero indicating which mode has been used to get the GPS coordinates. The variables are set in a SenML string for the GreenIoT cloud. Before publishing the SenML string to the MQTT Broker, the time of the measurement is calculated by subtracting the seconds received with the POST request from the server time. The sensor data is then published to the GreenIoT MQTT Broker for subscription. The HTTP API was set to store the data in a text file and in

the GreenIoT database.

6. Experiment results

6.1. Selection of bus routes

The goal was to identify two suitable routes for the project which cover the largest area in Uppsala city and include the three areas of interest where the stationary sensors are deployed. This project included a full deployment of one Waspmote Plug & Sense! sensor and the route recommendations for two Waspmote Plug & Sense! sensors owned by the GreenIoT project. The coverage percentage for a route was defined by how many pixels the route covers in the UL route map. As shown in Table 1, out of 21 bus routes, bus route 41 has the best estimated coverage over Uppsala city if only one bus route is to be selected. Out of 210 combinations of two bus routes, the combination of bus routes 42 and 12 gives the widest coverage over Uppsala city. For the deployment of the first sensor, bus route 22 was chosen eventually because of its long period of operation time per day. The second sensor was planned to be deployed on bus route 20 that covers a wider area and moves in more different parts of Uppsala.

The four areas where stationary sensors are deployed are considered as areas of interest for the GreenIoT project. None of the bus routes cover all the four areas of interest on their own. By adding the areas of interest into the route algorithm with a condition that at least one of the areas must be covered, the number of available routes drops from 21 to 15 routes. Considering the four areas of interest in the algorithm, the number of bus route combinations drops from 210 to 105 possible combinations. Table 2 shows the coverage percentage and the number of covered areas by a single bus line and multiple bus lines, respectively. Bus routes 9 and 42 form the best bus route candidate pair, which provides the largest coverage percentage but covers only three of the areas of interest.

6.2. Sensor performance

6.2.1. Communication performance

The goal of this experiment was to measure how well the sensor's 4G module and the two air quality sensors performed in data collection and communication. Table 3 presents, in numbers, how well the 4G module performed. Due to a corrupted data file, some of the data had to be retrieved from a backup database, which did not have the timestamp of the data record. An HTTP API was used to identify whether the record was obtained from the sensor's storage or was successfully sent to the HTTP API on the first try. Records that did not have the HTTP API timestamp had to be left out of the performance test.

Table 3 shows the total number of records in the database, which was almost forty thousand. Four thousand records were left out of the performance test due to not having the HTTP API timestamp or being corrupted into some unusable state. The Waspmote Plug & Sense! 4G module performance was acceptable thanks to the storage implementation in the sensor code. As shown in Table 3, 70% out of the total amount of the usable records reached the GreenIoT cloud on the first attempt. 30% of the messages would have been lost without the storage functionality for the 4G module unable to be connected to the service provider's mobile network. Based on the above results, we find the communication performance of the sensor satisfactory, in particular with the implementation of the storage function.

Table 1
Bus lines with best route coverage.

Single bus line	Coverage (%)	Two bus lines	Coverage (%)
41	20.00	42, 20	28.06
42	16.14	42, 12	26.71
11	15.17	42, 7	25.34
12	14.32	42, 14	24.90
20	14.28	42, 3	24.63
…	…	…	…
22	9.91	22, 20	20.07

Table 2
Bus lines with best coverage and most areas of interest satisfied.

Bus line	Number of satisfied areas of interest	Coverage (%)	Two bus lines	Number of satisfied areas of interest	Coverage (%)
1	3 of 4	7.13	9, 20	4 of 4	19.97
9	3 of 4	6.01	9, 5	4 of 4	16.52
5	2 of 4	11.00	9, 22	4 of 4	15.14
14	2 of 4	10.30	9, 42	3 of 4	21.50
22	2 of 4	9.91	9, 14	3 of 4	15.59

Table 3
4G module performance.

	Records	Percentage (%)
Received on the first try	24833	70
Received from storage	10453	30
Total records in Ref. Performance test	35286	100
Records in performance test	35286	90
Records without time stamp	3865	10
Records with corrupted data	18	0.046
Total amount of records	39169	100

6.2.2. Sensing performance

Table 4 shows the number and percentage of the zero and non-zero measurements. Forty thousand records were used, including the records retrieved from the backup database when measuring the sensor probe performance. The Waspmote Plug & Sense! and *CO* sensor probe achieved a robust performance as expected. Unfortunately, the NO_2 sensor probe did not meet the expected performance. As shown in Table 4, only a fraction of the NO_2 measurements exceeded zero value. According to Libelium technical support, the NO_2 sensor probe returns zero value if the concentration of the gas is too low for the sensor to measure it.

6.3. Analysis on data quality

6.3.1. CO sensor data

Fig. 8 shows all the data collected on *CO* concentrations from March to June 2017 with an 8-h mean, which is a standard mean used to calculate the AQI of the *CO* measurements. The average *CO* level in the period was $0.56 mg/m^3$ in the Uppsala city area. The figure shows that the data includes high peaks reaching over $10 mg/m^3$ and the highest peak even reaches over $20 mg/m^3$.

We planned to investigate whether the occasional high peaks of the *CO* measurements are due to sensor errors. Thus, we used a second Waspmote sensor, referred as the test sensor in the rest of the paper, to verify the measurements. Fig. 9 shows the *CO* concentrations against time. The blue line shows all the data points from the bus sensor on 19th June, 2017 between 14:45 and 17:08. The cyan line shows all the data points from the test sensor and the red line is a 4-min mean of the test sensor data.

Table 4
Success rate for the *CO* and NO_2 air quality sensors.

	CO	NO_2
non-zero measurements	38918	232
percentage	99%	1%
zero measurements	525	38625
percentage	1%	99%

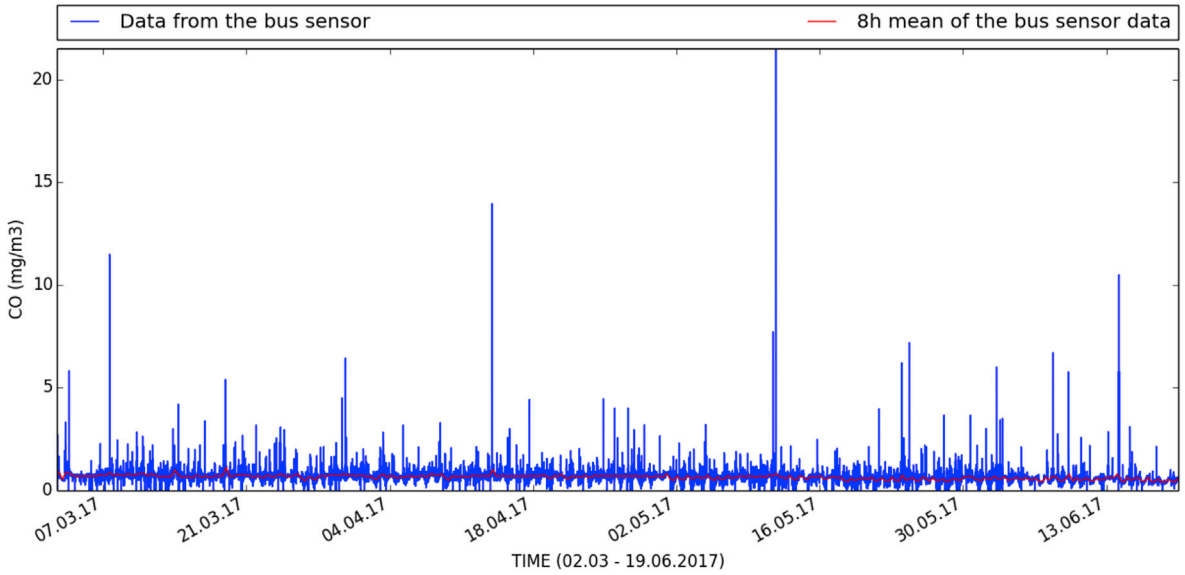

Fig. 8. All the *CO* data points from the bus sensor in the period of our experiment and the 8-h mean of the data that is used to calculate the AQI for *CO*.

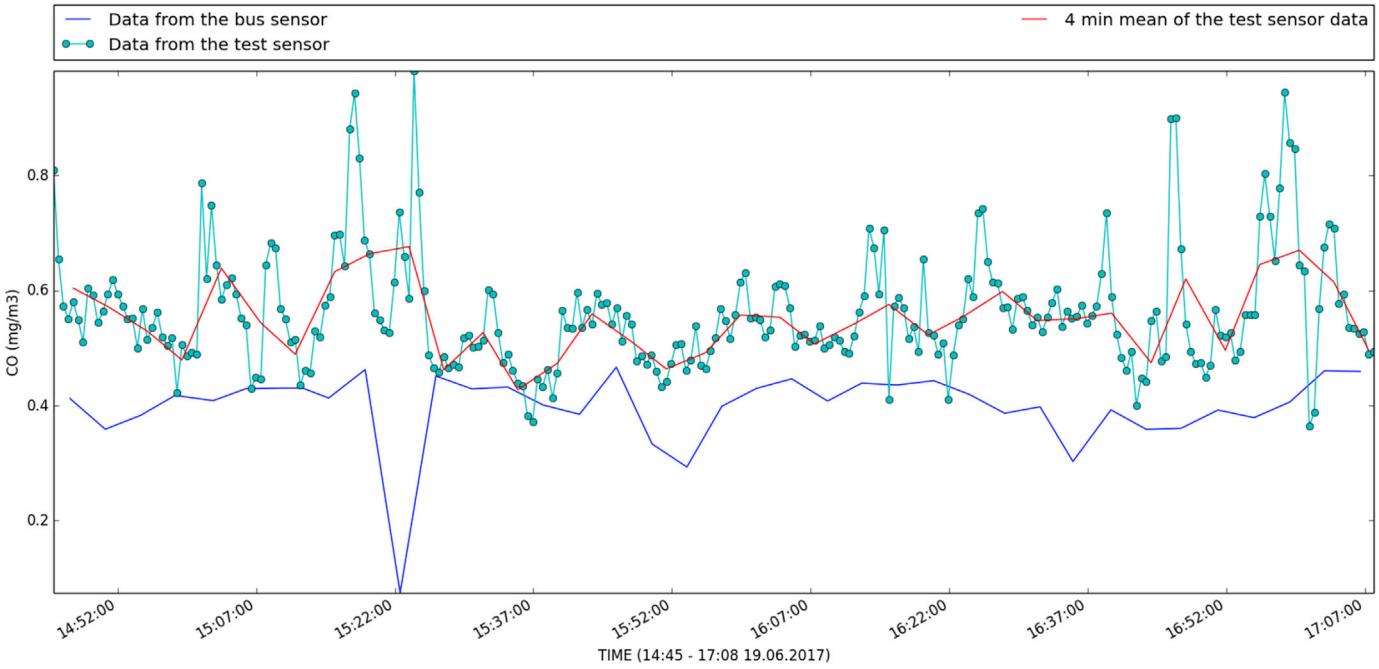

Fig. 9. Comparison between data from the bus sensor and more frequently measured data from the test sensor.

The data from the test sensor did not have peaks reaching over $10mg/m^3$. Unlike the data from the sensor on bus, the peaks in the data from the test sensor always contain more than one data points. Both sensors were calibrated by the manufacturer for low levels of *CO*. Measurements taken from the roof of the moving bus were constantly $0.2mg/m^3$ lower compared to the measurements taken by the stationary test sensor on the pedestrian level. At 15:22, the sensor on bus measured only $0.065mg/m^3$, which created a downward peak, while the test sensor measured some of its highest values at the same time. According to the GPS data of the bus sensor, there was only a distance of 500 meters between the two sensors at that time, which could point to an error in the bus sensor measurement seeing that the general *CO* concentration for both sensors were between $0.4mg/m^3$ and $0.6mg/m^3$.

We suspect that the occasional measurement errors of the bus sensor may be due to the Ultracaps cooling system effect. The Ultracaps cooling system is located on the bus roof next to the sensor, and it pushes out hot air when it is in use. The cooling system is activated when the Ultracaps are being used or re-charged by the energy received from the brake when the bus is stopping. Fig. 10 shows all the *CO* data points on *CO* concentration against temperature in week 11. The trend of the *CO* measurements seems to decrease slightly from lower temperature to higher temperature. The *CO* measurements have more variations during high temperature, though high *CO* values are still measured throughout the experiment. From these results, we see some correlation between the *CO* levels and the temperature. However, it is still not able to explain the occasional high measurements of *CO* from the bus sensor.

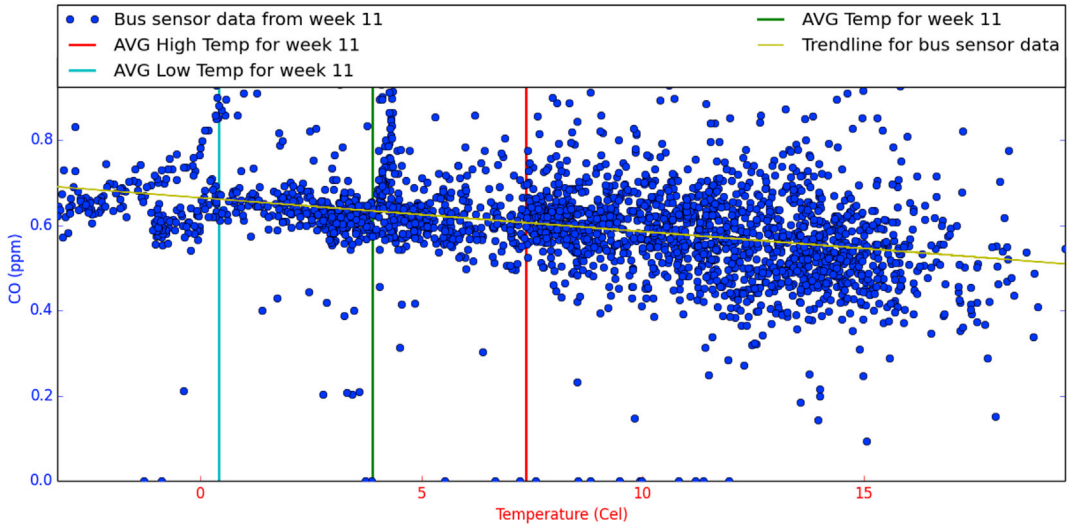

Fig. 10. Bus sensor data on *CO* concentration against temperature in week 11.

6.3.2. NO_2 data

The NO_2 data is analyzed in order to identify the reason for a large number of zero measurements in Table 4. Fig. 11 shows all the data collected on NO_2 concentrations against time. From the figure, it can be seen that the NO_2 sensor probe detected NO_2 concentrations throughout April and May.

A comparison dataset of the NO_2 data was acquired from SLB-analys, which is a division in Stockholm city's Environment and Health Administration management and collects data from air quality monitoring stations in Stockholm and Uppsala. Fig. 12 shows the NO_2 data of Kungsgatan provided by SLB-analys. The measurement unit of data from SLB-analys is micrograms per cubic meter ($\mu g/m^3$), while the measurements from the Waspmote Plug & Sense! sensor are in milligrams per cubic meter (mg/m^3). The highest NO_2 concentrations data measured by SLB-analys, over the period of our experiment, was $167.6\mu g/m^3$, which equals to $0.1676mg/m^3$. The accuracy point of the Waspmote Plug & Sense! NO_2 sensor probe used in this project was 0.1 ppm, which equals to $0.19mg/m^3$, indicating that all the measurements from the SLB-analys were out of the accurate sensitivity range of the Waspmote NO_2 sensor. This could explain the zero measurements from the Waspmote Plug & Sense! NO_2 sensor probe, for it is not sensitive enough for measuring the low NO_2 levels in Uppsala city.

7. Conclusions and future work

The development and deployment of an air quality monitoring system with movable sensors were conducted in Uppsala, Sweden. In the system, an air quality sensor was installed on the top of a public transport vehicle. The sensor measured NO_2, *CO*, temperature, humidity and ambient pressure. This work was a part of the GreenIoT project that also included other stationary sensors deployed in the city center of Uppsala. The *CO* sensor probe performed as expected, but the NO_2 sensor had only a one-percentage success rate in measuring non-zero concentration, which is likely due to its low sensitivity. The data was delivered using 4G mobile network, which gave 70% success rate without re-sending. This success rate reached 100% with the storage and re-transmission functions implemented.

Future work may include further development of the sensor program and analysis of the sensor data quality. To ease the updating of the sensor

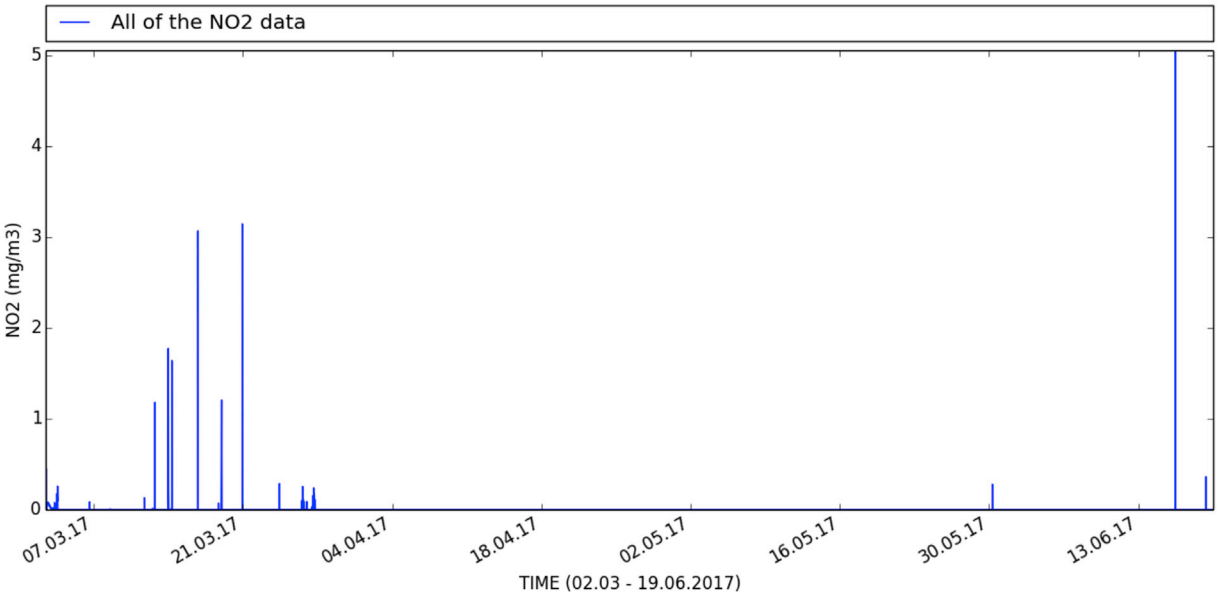

Fig. 11. All the NO_2 concentration data points collected by the bus sensor.

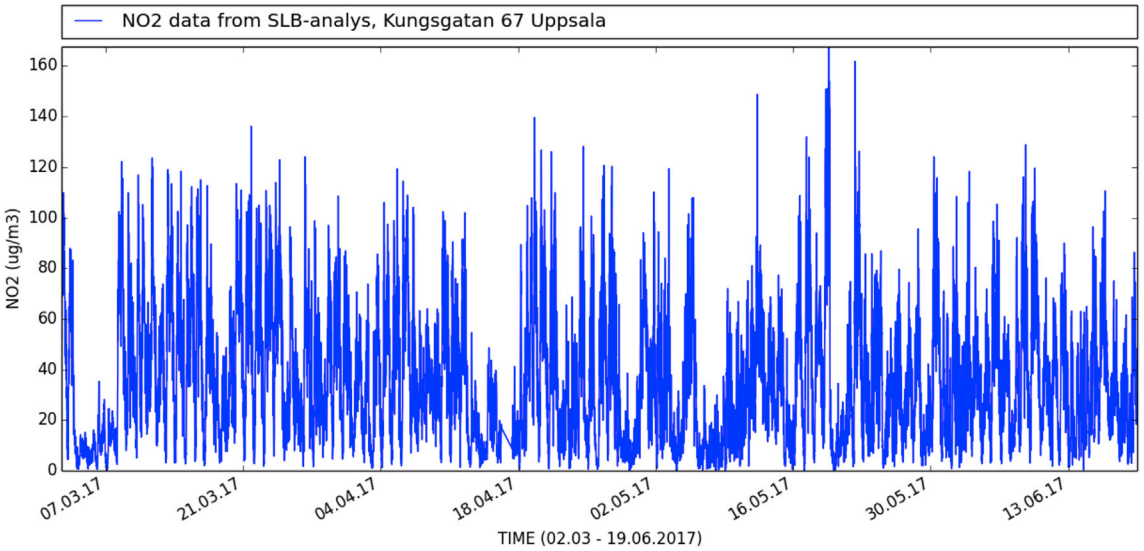

Fig. 12. NO_2 data from SLB-analys over the period of this experiment.

codes, the Over The Air Programming (OTAP) function could be investigated and implemented. In addition, more experiments could be conducted to study how the moving speed, the wind speed, and the cooling system of the bus may affect the sensor measurements.

Acknowledgements

This work was supported by the GreenIoT project grant (2015-00347) from VINNOVA, Sweden. We would also like to thank Tommy Rydbeck and his colleagues from Gamla Uppsala Buss AB for the technical support and collaboration, which made the experiments in this work possible.

References

[1] H. Kopetz, Internet of Things, Springer US, Boston, MA, 2011, pp. 307–323.
[2] P. Vlacheas, R. Giaffreda, V. Stavroulaki, D. Kelaidonis, V. Foteinos, G. Poulios, P. Demestichas, A. Somov, A.R. Biswas, K. Moessner, Enabling smart cities through a cognitive management framework for the internet of things, IEEE Commun. Mag. 51 (6) (2013) 102–111.
[3] J. Jin, J. Gubbi, S. Marusic, M. Palaniswami, An information framework for creating a smart city through internet of things, IEEE Internet of Things Journal 1 (2) (2014) 112–121.
[4] A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of things for smart cities, IEEE Internet of Things Journal 1 (1) (2014) 22–32, https://doi.org/10.1109/JIOT.2014.2306328.
[5] C. Zhu, V.C.M. Leung, L. Shu, E. Ngai, Green internet of things for smart world, IEEE Access 3 (2015) 2151–2162.
[6] B. Brunekreef, S.T. Holgate, Air pollution and health, Lancet 360 (9341) (2002) 1233–1242.
[7] B. Ahlgren, M. Hidell, E. Ngai, Internet of things for smart cities: interoperability and open data, IEEE Internet Computing 20 (6) (2016) 52–56, https://doi.org/10.1109/MIC.2016.124.
[8] A. Kadri, E. Yaacoub, M. Mushtaha, A. Abu-Dayya, Wireless sensor network for real-time air pollution monitoring, in: International Conference on Communications, Signal Processing and Their Applications, ICCSPA), 2013, pp. 1–5.
[9] V. Hejlová, V. Vozenilek, Wireless Sensor Network Components for Air Pollution Monitoring in the Urban Environment : Criteria and Analysis for Their Selection, Wirel. Sens. Netw. 5 (12) (2013) 229–240.
[10] C. Antonopoulos, F. Kerasiotis, C. Koulamas, G. Papadopoulos, S. Koubias, Experimental evaluation of the waspmote platform power consumption targeting ambient sensing, in: 2015 4th Mediterranean Conference on Embedded Computing (MECO), 2015, pp. 124–128, https://doi.org/10.1109/MECO.2015.7181882.
[11] W. Fuertes, D. Carrera, C. Villacs, T. Toulkeridis, F. Galrraga, E. Torres, H. Aules, Distributed system as internet of things for a new low-cost, air pollution wireless monitoring on real time, in: 2015 IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), 2015, pp. 58–67, https://doi.org/10.1109/DS-RT.2015.28.
[12] O. Saukh, D. Hasenfratz, A. Noori, T. Ulrich, L. Thiele, Demo-abstract: route selection of mobile sensors for air quality monitoring, in: 9th European Conference on Wireless Sensor Networks (EWSN 2012, 2012, pp. 10–11.
[13] L. Gugliermetti, M. Sabatini, G.B. Palmerini, M. Carpentiero, Air quality monitoring by means of a miniaturized sensor onboard an autonomous wheeled rover, in: 2016 IEEE International Smart Cities Conference (ISC2), 2016, pp. 1–4, https://doi.org/10.1109/ISC2.2016.7580868.
[14] C. Jennings, Z. Shelby, J. Arkko, A. Kernen, C. Bormann, Sensor Measurement Lists (SenML), RFC 8428, Aug. 2018, https://doi.org/10.17487/RFC8428. URL, https://rfc-editor.org/rfc/rfc8428.txt.
[15] Libelium, Waspmote plug & sense technical guide, Tech. rep (2017). http://www.libelium.com/downloads/documentation/waspmote_plug_and_sense_technical_guide.pdf.
[16] M. Truck, B. AG, The future will be riding green. man lions city hybrid, Tech. rep. (2015) 1–12. http://www.neoplan.se/uploads/files/hybrid_2015_en.pdf.