Journal of Computer Science and Information Technology
December 2019, Vol. 7, No. 2, pp. 37-48
ISSN: 2334-2366 (Print), 2334-2374 (Online)

Published by American Research Institute for Policy Development
DOI: 10.15640/jcsit.v7n2a5
URL: https://doi.org/10.15640/jcsit.v7n2a5

Smart Building Temperature Monitoring System using Matplotlib

Xinzhou Wei[1] , Xiaowen Zhang[2] & Li Geng[3]

Abstract

Wireless sensor network (WSN) has been widely adopted by applications of Internet of Things (IoT). In this paper, we present a practical design of real time temperature monitoring system for smart building by using XBee based mesh WSN. The proposed system collects temperature data from wireless sensors and dynamically displays those data using Matplotlib with Python. The data was recorded by the SQLite3 database simultaneously for cloud usage. Moreover, we evaluated the link quality of the wireless transmission in our system. Experimental results demonstrated that the proposed real time smart building temperature monitoring system is a cross-platform and low cost system for IoT applications. It could be easily applied to single board computer systems such as Raspberry Pi, Jetson Nano, or LatteePanda to further reduce the cost.

Keywords: Data visualization, Internet of Things, Matplotlib, Smart building, Wireless sensor network, XBee

1. Introduction

Internet of Things (IoT) and its applications are changing our daily lives in many fields, ranging from wearable smart gadgets to home appliances. Wireless sensor network (WSN) plays a very important role in the infrastructure of IoT. It has been applied to the fields of smart cities [Mitton et al., 2012], street light control [Sunehra et al., 2017], smart building management [Ghayvat et al., 2015; Liang et al., 2017; Minoli et al., 2017; Zanella et al., 2014], environmental and air pollution monitoring [Nagaraj et al., 2017; Ray 2015], energy monitoring [Grosso et al., 2018; Liu et al., 2012; Ray 2016], health monitoring [Chauhan et al., 2016; Garrido et al., 2019; Othman et al., 2014], and intelligence & precision agriculture [Calderón-Córdova et al., 2018; Davcev et al., 2018; Granda-Cantuña et al., 2018; Ma et al., 2018; Medeiros et al., 2018; Nooriman et al., 2018; Onibonoje et al., 2017; Sriploy et al., 2018]. Monteiro et al. (2018) proposed a forecast model for indoor temperature prediction in an IoT scenario composed of a home appliance. It integrated information from own and external sensors and performed a forecast on the ambient temperature. A comprehensive study of the state-of-the-art research on machine learning, architectural, technical aspects of the smart building management systems has been reviewed by Djenouri et al. (2019). Dayarathna (2019) further presented a detailed comparison of current general IoT development platforms which include Appcelerator, Amazon Web Services(AWS), Bosch, IBM, ParStream, ThingsWorx, Xively, and ThingsBoard.

Data that is collected by wireless sensors on the "things" in IoT includes temperature, humidity, speed, position, force, or the density of some gases of the objects, vehicles, and the environment. Displaying the data in a more appealing and interactive way for the user is a challenge task for data scientist and data engineers. Data visualization is one of the most effective tools to interpret big data. It allows human being to better understand the nature of the dataset, as well as conveys the proper message to non-technical viewer. A lot of data visualization tools for WSN have been developed in past years [Castillo et al., 2008; dAuriol et al., 2010; ElHakim et al., 2010; Parbat et al., 2010; Ravichandran et al., 2016]. Here we briefly introduce some of the most popular data visualization tools used in the WSN field.

[1] Department of Electrical and Telecommunications Engineering Technology, New York City College of Technology, 300 Jay St, Brooklyn, NY 11201. Email: xwei@citytech.cuny.edu
[2] Department of Computer Science, City University of New York/College of Staten Island, 2800 Victory Blvd., Staten Island, NY 10314. Email: xiaowen.zhang@csi.cuny.edu
[3] Department of Electrical and Telecommunications Engineering Technology, New York City College of Technology, 300 Jay St, Brooklyn, NY 11201. Email: lgeng@citytech.cuny.edu

The professional software like MOTE-VIEW [Tuton, 2005] provides many powerful functions for user but it is too sophisticated for a small scale WSN. SpyGlass [Buschmann et al., 2005] is very powerful and flexible in modular visualization application for WSN. It allows users to develop their own special visualization plug-ins which could be added in SpyGlass. However, it requires the user to fully understand the architecture of SpyGlass and its visualization components. Otherwise, there is no guaranty for the compatibility of user's plug-in component. NetTopo [Shu et al., 2008] is a platform independent, flexible WSN simulation and visualization tool. It also could be used as a test-bed for some devices such as camera and Bluetooth based wireless sensors, but it provides more functions on simulation than visualization. Octopus [Jurdak et al., 2011] is a modular visualization and control tool. It provides GUI to display the topology of WSN. It also provides live data plot for users. However, if users want to test their own node program, Octopus won't fully support it. Syafrudin et al. (2018) proposed a big data processing platform which utilizes Apache Kafka as a message queue, Apache Storm as a real-time processing engine, and MongoDB to store the sensor data for automotive manufacturing. Though the system is sufficiently efficient to monitor the manufacturing process, it is very costly. Thus, there is a need to develop a low cost, cross-platform, efficient implementation visual tool for non-programming engineers in order to display data collected by wireless sensors for real time applications.

This paper introduces an efficient solution of data visualization scheme using Matplotlib in Python for WSN. The data collected by wireless sensors are transmitted via routers in WSN and then forwarded to the personal computer from the coordinator via serial port. Then, the data are animated visually in different real time plotting created by Matplotlib command based on their locations or MAC addresses. A record of data is created by SQLite database system simultaneously according to the corresponding MAC address of the sensor. The timestamp is added to the record at the same time. SQLite is a user-friendly database engine which is self-contained, server-less, zero-configuration and transactional. It is very fast and lightweight with the entire database stored in a single file, requiring no configuration and stores information in ordinary disk files. Comparing with other open source database like PostgreSQL and MySQL (which are suitable for complex operations), SQLite is a popular choice as a database to support small to medium-sized websites. It is also used in a lot of applications as internal data storage. The Python standard library includes a module called "sqlite3" which is intended for working with this database. This module is a SQL interface compliant with the Database-Application Programming Interface (DB-API) 2.0 specification (https://www.pythoncentral.io/introduction-to-sqlite-in-python/).

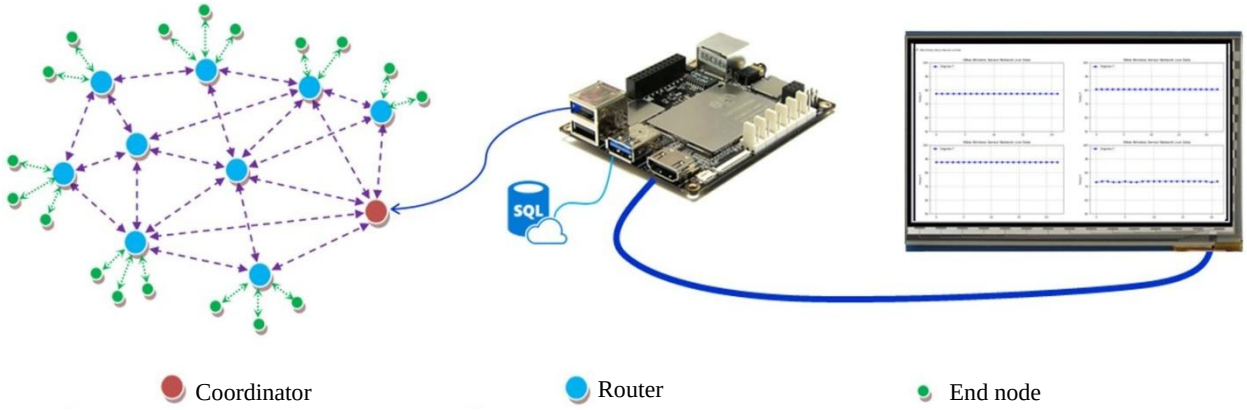

Figure 1. A typical XBee wireless sensor network system.

The XBee based wireless sensor network is composed of a coordinator and many routers as demonstrated in Fig. 1. The role of the coordinator is to get all data forwarded by routers of the WSN and send them to the computer via serial port. The router of an XBee based WSN is in charge of forwarding the data collected by the sensors on the end node to other routers or to the coordinator via a mesh network. In addition, all XBee modules in the same WSN should contain the same Personal Area Network (PAN) ID number. Detailed configuration steps of XBee coordinator and router could be found in Faludi's work [Faludi, 2010].

Digi International provides free application named XCTU (XBee Configuration and Test Utility) to program, configure, troubleshoot, and manage the XBee based mesh network. The XCTU software can be downloaded from the Digi's website (http://www.digi.com/products /wireless-wired-embedded-solutions/ZigBee-rf-modules/xctu). The total cost of the proposed system is in the range of hundreds of dollars, including a personal computer, XBee routers/coordinator, and temperature sensors.

The remainder of this paper is organized as follows: Section 2 describes the topology of wireless sensor network and the configurations of XBee modules. Section 3 presented the algorithm and flowchart of data acquisition and conversion from wireless sensors. Section 4 provides the usage of Matplotlib plotting functions in Python. The SQLite database system recording methods are described in Section 5. The experimental results are presented in Section 6. The conclusion and future work are provided in Section 7.

2. Topology of the WSN and XBee Module Configuration

2.1. The wireless sensor network topology

To display the topology of our XBee based WSN, we use the XCTU's Network function to discover and visualize the topology. There is a "Network working mode" button shown on the top of XCTU user interface. When this button is clicked, XCTU scans and displays the logical connections and link quality of the entire topology of the WSN as shown below.

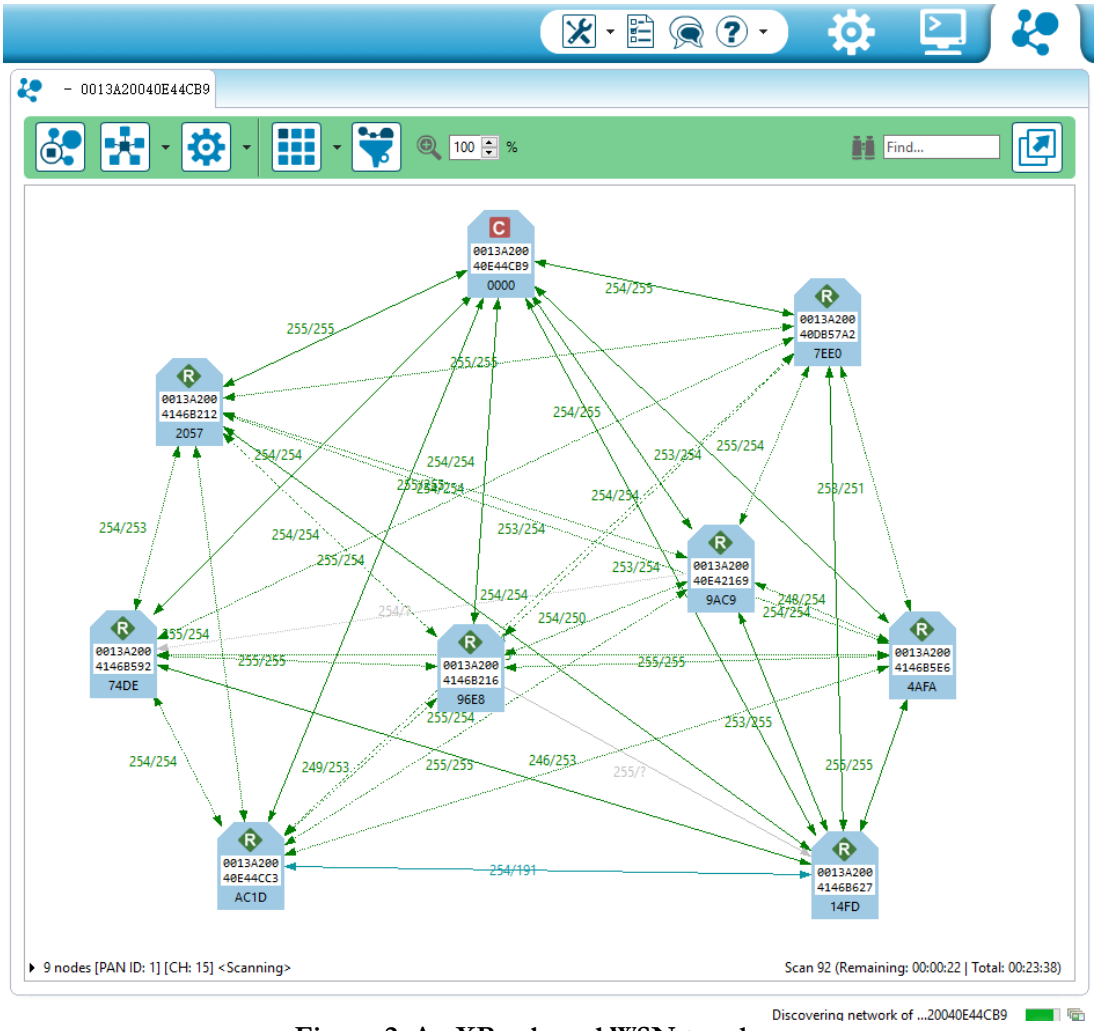

Figure 2. An XBee based WSN topology.

Fig. 2 shows the topology of our WSN detected by XCTU. The router is represented by letter 'R' in blue color and the coordinator is represented by letter 'C' in red color in the XBee icons.

To examine the detailed connection information for a specific XBee module, we click any XBee icon in XCTU network window, a connection table would pop-up and display all of the connections with that XBee module. There are two numbers that indicate the packets sending and receiving between two XBee modules for each link. If the number is 255/255, it represents that all packets sending to another module get all responses, i.e., the link is health. It will be highlighted in green color. On the other hand, if the number is 255/?, it means that the outgoing 255 packets don't get response and the link is down. So the link between those two modules is not health. We will label the color of the link dim grey.

2.2. XBee module configuration

To configure the XBee module as a router, firstly, we mount the XBee module on an adapter called XBee explorer board. Developers can obtain the XBee explorer board from the Sparkfun Electronic Inc. or Adafruit Inc. Secondly, we attach a mini USB cable from this adapter and connect it to a PC. There is an onboard FT231X USB-to-Serial converter that translates data between XBee and the PC. Finally, we start the XCTU which will automatically detect the XBee module connected via USB port. In our XBee router, the value of pin DIO0 (Digital Input Output port 0) is configured as the input port of ADC (Analogue Digital Converter) which has a value of 2.

To configure the XBee module as a coordinator, the XBee module has been set to API (Application Programming Interface) mode and the PAN ID of the coordinator must have the same number with the XBee routers in the WSN. In our WSN system, the pin of DIO0 for the XBee coordinator is configured as a digital input port, which has a value of 3. All XBee modules on a WSN must have the same PAN ID number. Otherwise they could not be able to communicate with each other. In our WSN system, the PAN ID is configured as 1 for the simplicity of demonstration purpose. More details on XBee module configuration could be found in our research paper [Wei et al., 2017].

3. Data Acquisition and Conversion

In order to obtain data from wireless sensor via XBee modules in WSN, our system need import Python packages such as XBee, ZigBee, and serial packages before we start collecting data from sensors. For XBee modules, the baud rate is configured as 9600 bit/s. In order to save the data collected by different XBee modules, we create several lists as global variables that will be valid for the whole program. The database named "MyWSNData.db" is created by the command **conn = sqlite3.connect("MyWSNData.db")** and **curs = conn.cursor().**

The command

ser = serial.Serial(SERIALPORT, BAUDRATE)
xbee = ZigBee(ser)

will communicate with XBee modules via serial ports.

After the correct serial port is successfully detected, our system will start to read the data from the corresponding serial port. The data of wireless sensors will be read and saved in the variable

responseData = xbee.wait_read_frame().

The data in XBee data frame is a dictionary which includes the key values, such as "id", "source address long", "source address", and "sample values" as indicated below:

```
{ 'id': 'rx_io_data_long_addr', 'source_addr_long': b"\x00\x13\xa2\x00AF\xb6'",
  'source_addr': b'\x10\xf5', 'options': b'\x01', 'samples': [{'adc-0': 837}] }
```

Our system uses the first 250 data frames to collect MAC addresses of XBee modules on the entire WSN. In order to reduce the redundancy of the MAC addresses, we move the members of MAC addresses from a List[ ] to a Set( ) which is unordered, no duplication data structure. In other words, we keep the unique MAC addresses in our WSN. Finally the members of the Set( ) is sent to a new MAC address list which includes all XBee modules without any redundancy.

Next, the data of the room temperature is collected from the serial ports. A voltage value is obtained and converted to the corresponding temperature according to the following equation:

$$temperature = (volt\_average / 1023.0 * 1.2 * 3.0 * 100) - 273.15, \quad (1)$$

where volt_average is the voltage value obtained from the temperature sensor. An ADC in XBee converts the voltage value to a digital value. The value 1023 is 0x3FF in binary format which represents the maximum value of the ADC reading. The converted temperature value is represented by degree Kelvin. Temperature in degree Celsius is obtained by subtracting 273.15 from degree Kelvin. For more detailed description, please refer Faludi's website at https://www.faludi.com/bwsn/tmp36-instructions-simple-sensor-network.

These temperature values will be appended to a different temperature list according to their MAC address. The values in the SQLite3 database will be updated simultaneously by calling an update database function. This procedure will be executed in a forever loop in which the data is kept receiving from the serial port till the user stops system from keyboard. A more detailed procedure is shown in Fig. 3.

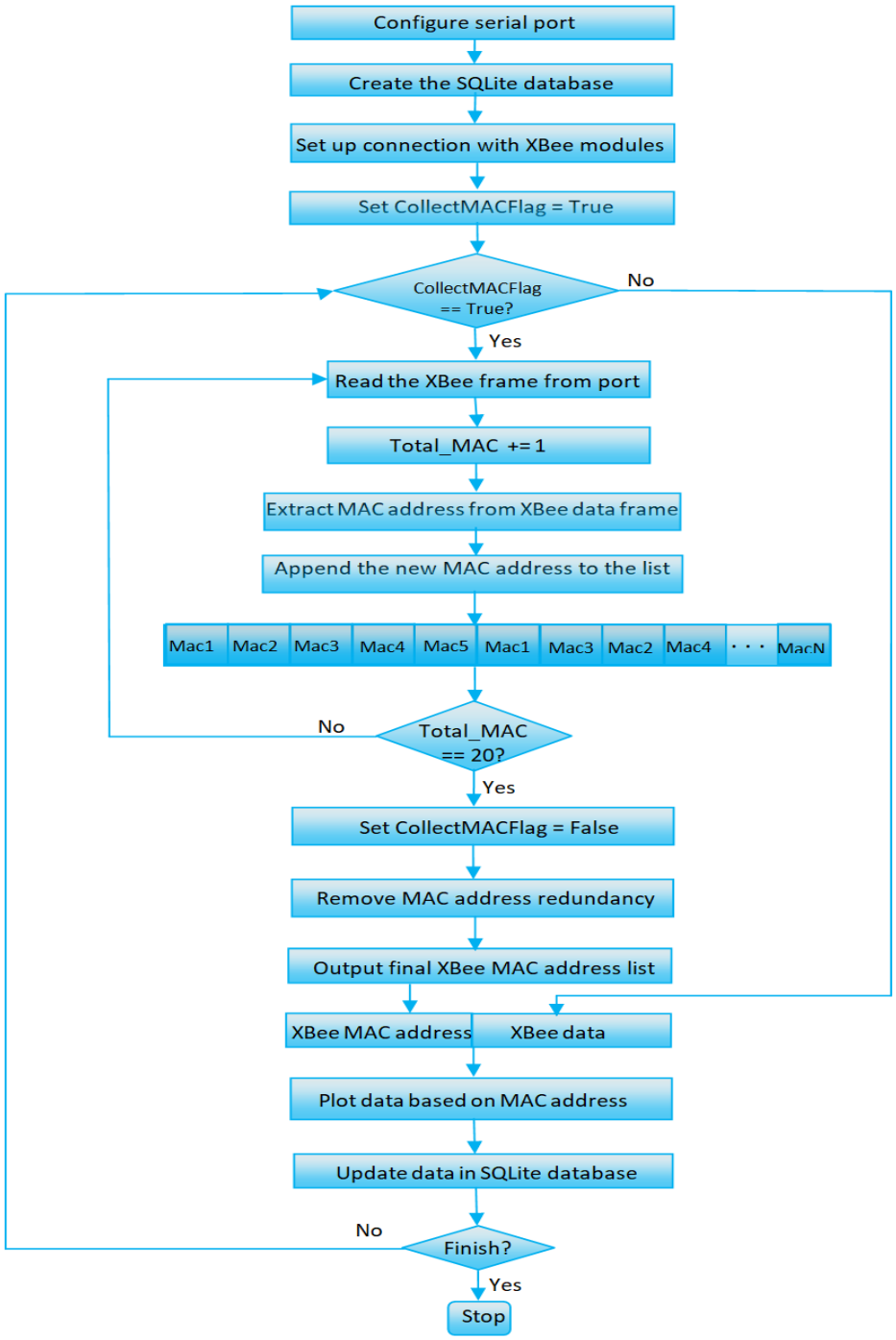

Figure 3. Algorithm flowchart of the XBee based WSN.

4. Data Plot using Matplotlib

Each temperature data in a list represents the data collected by the wireless sensors in real time. We have a limitation of the list size. When the data exceeds the size of the list, we apply the pop function on the list in which we remove the oldest data from that list. Based on the total sensors in the WSN, we can plot the figures in sequential order vertically, i.e. one figure in each row. We also can apply two or three figures in each row by applying Matplotlib function plt.subplot(MNI), where M represents number of rows and N represents the numbers of columns. The value I represents the index of current figure. The drawnow function and plot function in different subplot figures will be refreshed periodically.

To distinguish the locations of the sensors, we demonstrate the location information in the "xlabel" which is shown at the bottom of each plot. Fig. 4 shows a dynamic plot of live data in tempF list from a specific sensor. The newly collected data is shown on the right of the display window. The total amount of data is within the length of the list temF.

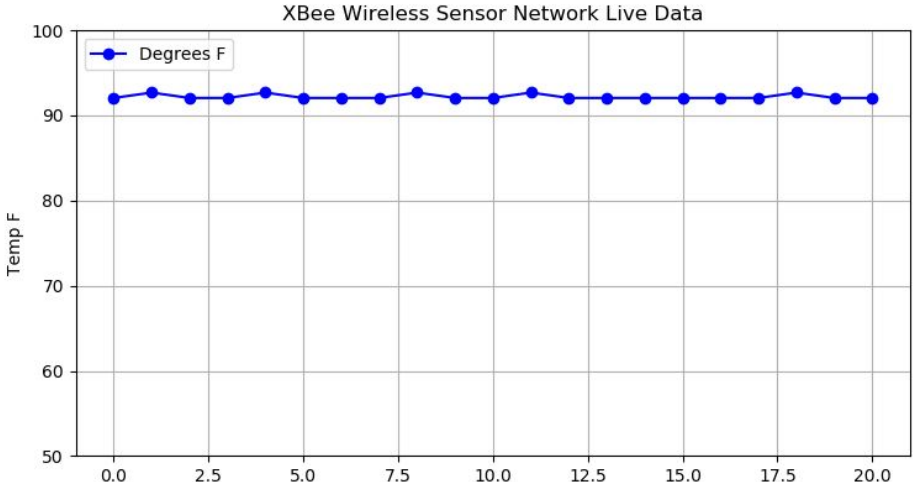

Figure 4. Values in tempF list.

5. Data Record in the SQLite Database

In order to keep the data log or upload the data to the cloud, we utilize the SQLite, a simple database, to record the temperature data with its MAC address and timestamp in the SQLite database. SQLite is a self-contained, server-less, full-featured SQL database engine. It has a very small footprint with an efficient usage of memory and storage space. It is highly reliable and doesn't need the maintenance from a professional database administrator. Therefore, SQLite is an excellent choice for the database engine in mobile devices such as smart phones, smart watches and other wearable electronic gadgets. Using a SQLite database with Python, we can build a cross platform database and shared the data seamlessly among smart devices.

Here are the code snippets that create a database to hold real time temperature from our WSN system. To use the SQLite database in our system, we need to add"import sqlite3" statement to our python script. In our WSN system, we created a database file called "**MyWSNdata.db**".

We use the function sqlite3.connect to connect to the database. Next we create a cursor object, which allows you to interact with the database and add records.

```
conn = sqlite3.connect("MyWSNData.db")
curs = conn.cursor()
```

SQLite3 only supports five data types: null, integer, real, text and blob. Next, we apply the SQL syntax to create a table named "**dataToPlot**" with two text fields "**mac_address date**", "**time**" and one real number field "**temperature**".

```
def create_table():
    curs.execute("CREATE TABLE IF NOT EXISTS dataToPlot(
                    mac_address TEXT, temperature REAL, date_time TEXT)")
```

The following code will insert data into our new table:

```
def insert_data(macAddress, temp):
    unixValue = time.time()
    dateTime = str(datetime.datetime.fromtimestamp(unixValue).strftime(
                    '%Y-%m-%d %H:%M:%S'))
    curs.execute("INSERT INTO dataToPlot (mac_address, temperature, date_time)
                VALUES (?, ?, ?)", (macAddress, temp, dateTime))
    conn.commit()
```

Here we call the SQL command "INSERT INTO" to insert a record into our database. In our curs. execute function, we use question marks (?) instead of string substitution (%s) to insert the values. To save the record to the database after insert operation, we need confirm our operations by calling conn. commit function.

In addition, we can read the data in SQLite by running the queries and then let the cursor object to execute the SQL command. Here are the code snippets for reading the data from the database:

```
def read_data_from_db(sensor):
    curs.execute('SELECT temperature FROM dataToPlot WHERE mac_address= ?',(sensor,))
    data = curs.fetchall( )
```

The query we execute is a "SELECT" command in SQL, which means that we want to select all the records that match the "mac_address" of a sensor. Finally, we execute the SQL command and use fetchall function to return all the results.

6. Experiment Results

6.1. Matplotlib plot results

As shown in Fig. 5, we diplayed real time temperature dynamically from four XBee sensors. The xlabel is the room number and the corresponding XBee MAC address. The dotted line in each plot represents the temperature value saved in the temperature list. When the total number of the dots exceed 20, the older values are removed from left side of the plot.

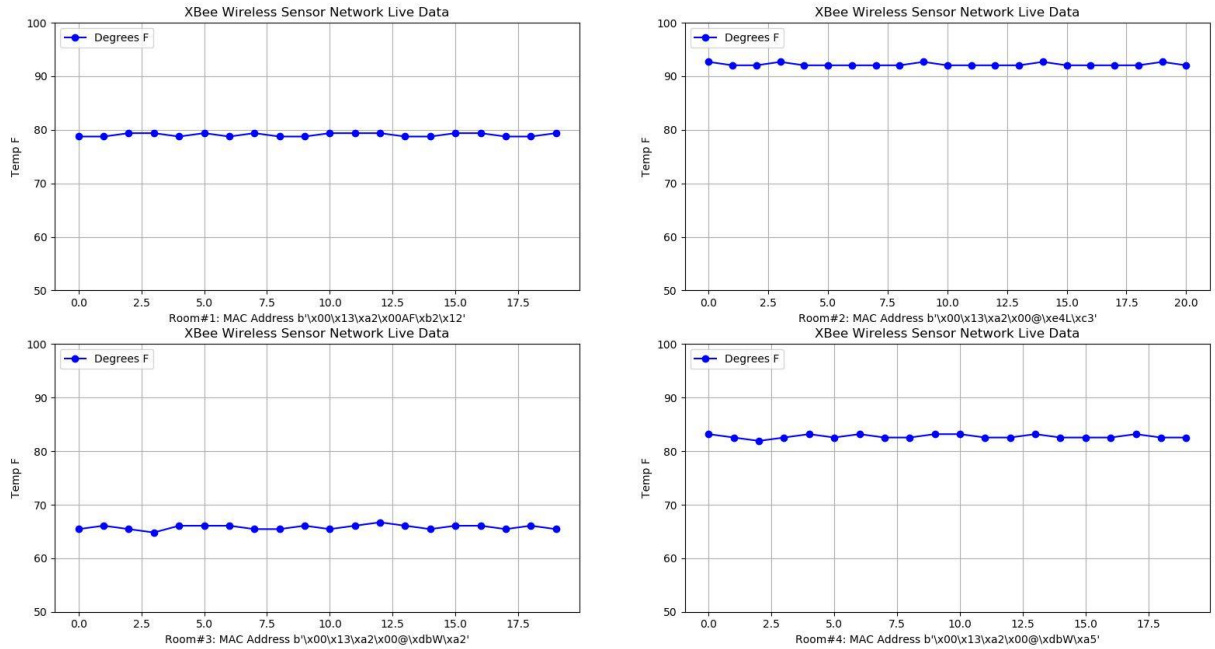

Figure 5. Live data visualization in matrices format using Matplotlib.

Matplotlib also provides a variety of plotting tools for user. All these plotting functions are command style functions that make Matplotlib work like MATLAB. Figure 6 shows the histogram style of live data in our XBee based WSN.

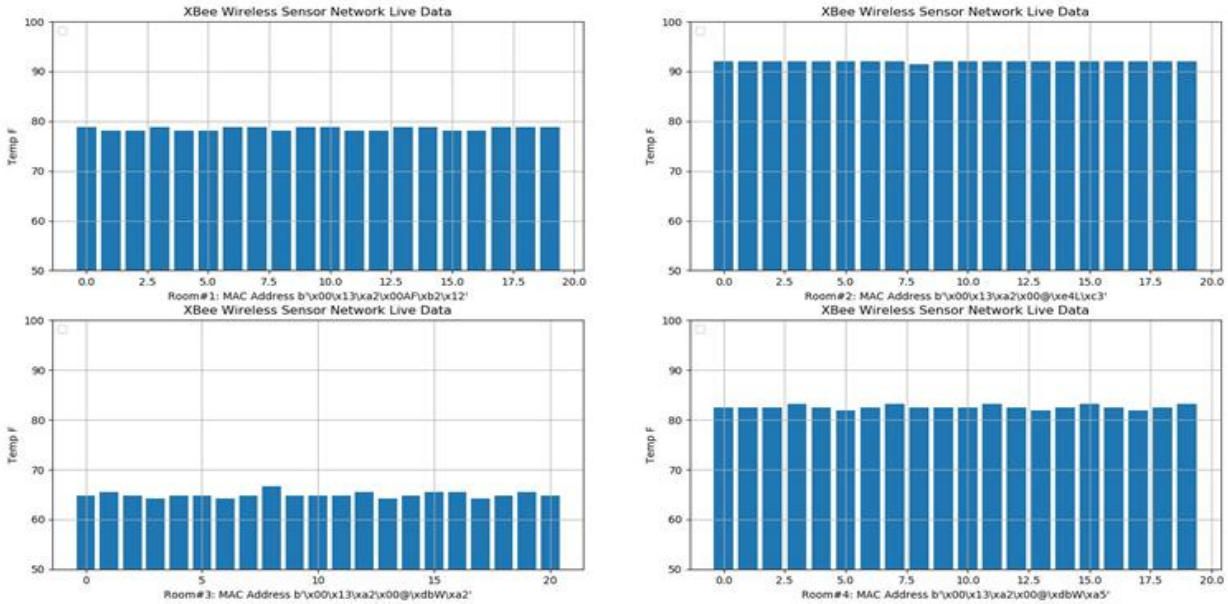

Figure 6. Live data visualization in histogram format using Matplotlib.

If the total number of the sensors are not even number, user can display data in vertical style with one column plot as shown in Fig. 7. The value on the right side is the latest value. The value on the left side is the oldest value and will be removed when the total amount of values exceed 20.

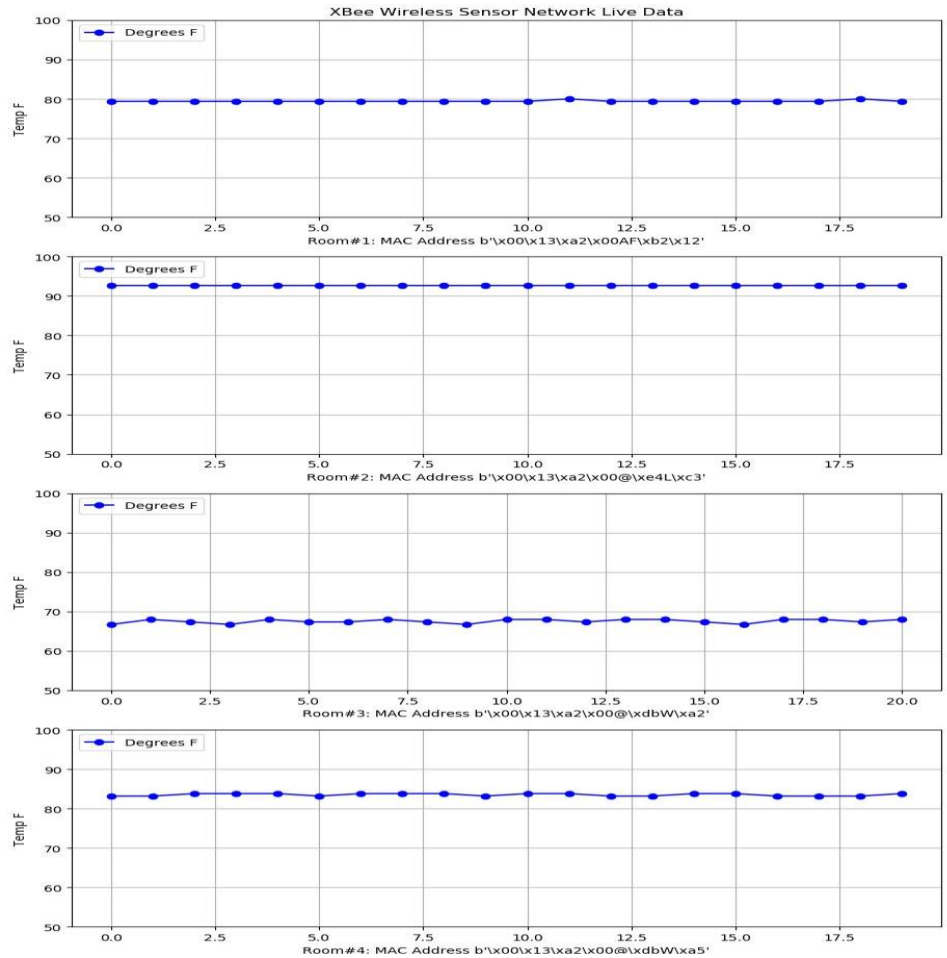

Figure 7. Live data visualization in sequential order using Matplotlib.

For the smart building management application, a weekly or monthly average temperature data might help administration team to optimize the energy consumption. Fig. 8 shows the weekly average temperature value collected from sensors.

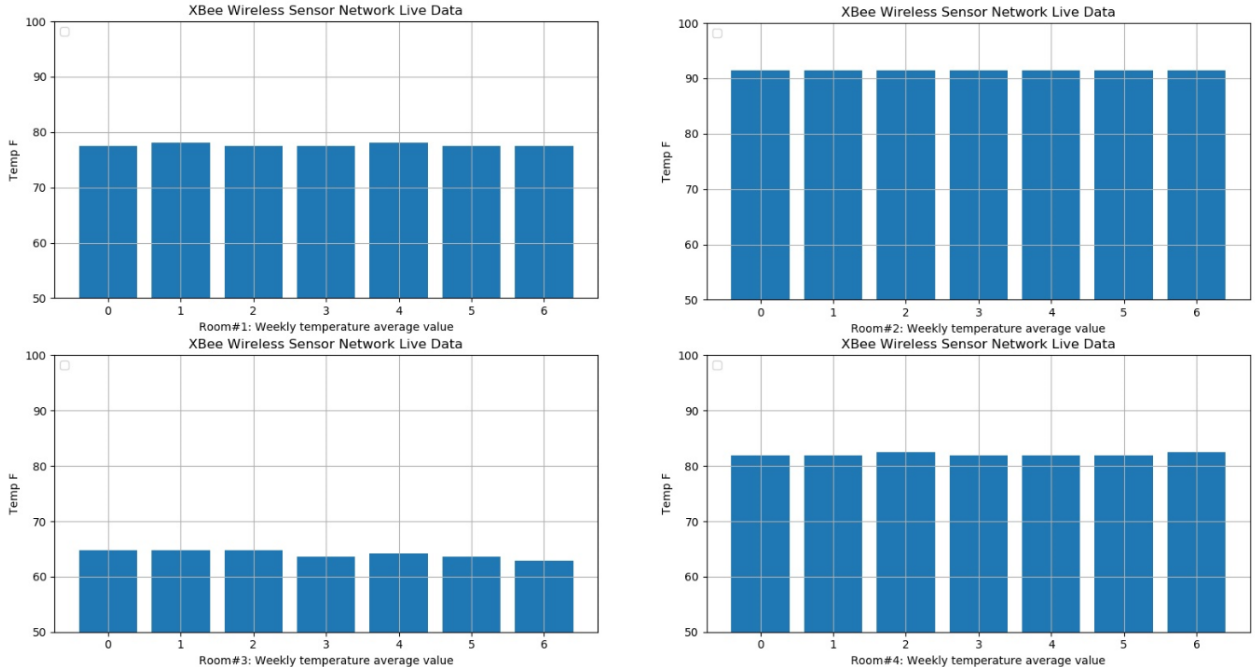

Figure 8. Weekly average temperature value of each sensor, where 0-6 indicates seven days in a week.

6.2. SQLite database results

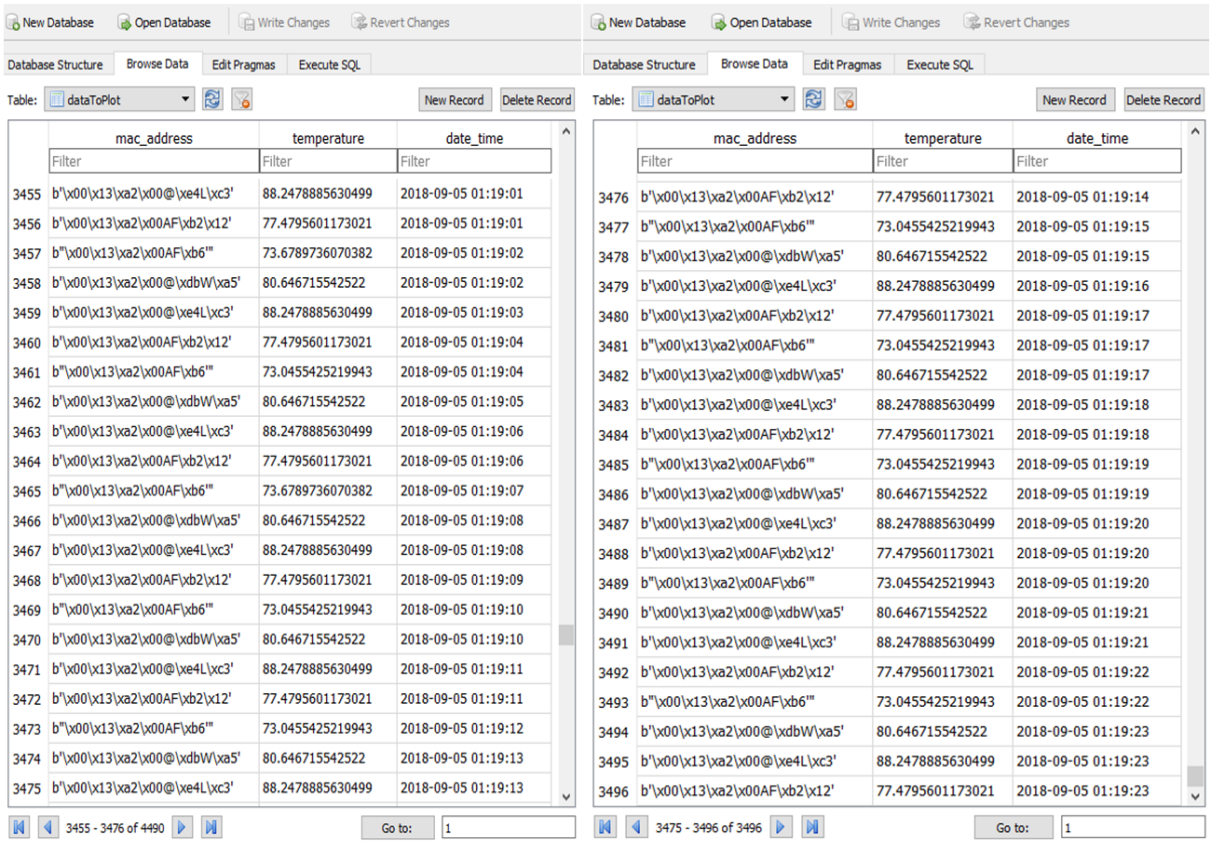

Figure 9. Data record by the SQLite3 database.

The temperature values are also recorded in a database created by the SQLite3. The results shown in Fig. 9 are the data recorded by the SQLite3 and displayed by the SQLite browser. We kept the MAC addresses of XBee nodes, temperature values, date and time in the table.

These data could be uploaded on the cloud and accessed by the users via Internet.

6.3 Data analysis on link quality

In order to evaluate the quality of the wireless transmission of our XBee based WSN, we conducted a field test on the performance of four nodes in our system. As shown in Fig. 2, the quality of the wireless connection between two nodes were displayed by two numbers over the link which is called Link Quality Indicator (LQI). The LQI shows a number between 0 to 255 where 0 represents the weakest connection and 255 represents the strongest connection. In this work, we measured the LQI values for each node based on their distance to the XBee coordinator from 0 to 35 meters. As shown in Fig. 10, the quality of the link decreased in reverse proportional to the distance. It is noted that the decreasing of the quality may be due to the signal attenuation through room furniture or building walls.

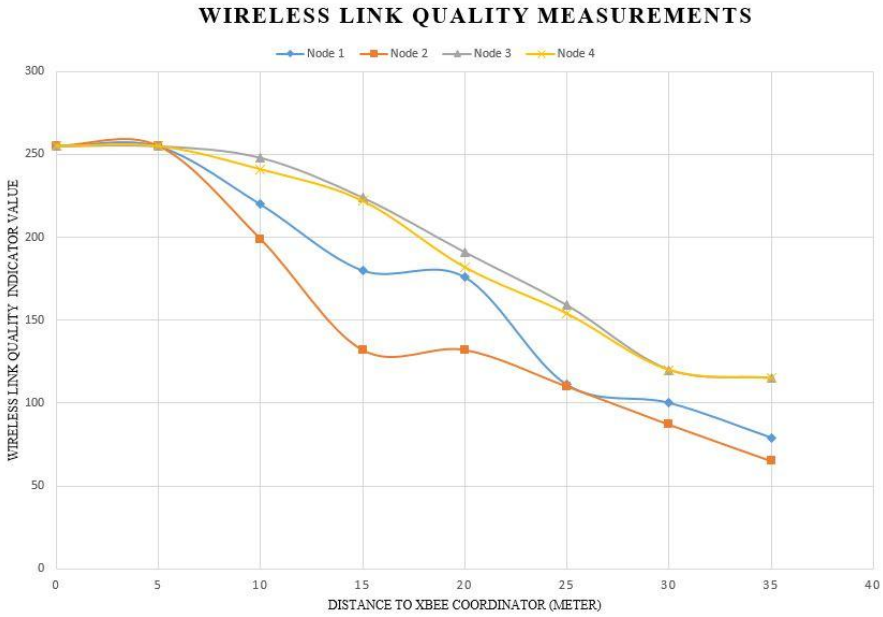

Figure 10. Wireless link quality measurements.

7. Conclusion

In this paper, we presented a novel low cost solution to real time temperature monitoring system for smart building using XBee based WSN. The experiment results demonstrated that data collected by the wireless sensors could be dynamically visualized by Matplotlib plotting. The data was also recorded by the SQLite3 database simultaneously for cloud usage. Moreover, we evaluated the link quality of the wireless transmission in our system. The system was implemented purely using Python which is an open source, cross platform language. The total system cost could be further reduced by adopting the single board computer like LattePanda, ODROID-XU4, and Raspberry Pi. For our future work, the proposed system can be extended to collect data in the smart building with multiple tasks, such as humidity, carbon monoxide, and electricity consumption, where other open source databases, like PostgreSQL or MySQL, would be a better choice to handle complex tasks than SQLite. In addition, if we extend our system to a large scale WSN (e.g. a university campus with 500 rooms), the effect of bandwidth limitation shall be evaluated to ensure the quality of data transmission.

Acknowledgments

This work was partly supported by CUNY GRTI grant round 20 and PSC-CUNY grant #61163-00 49, jointly funded by the City University of New York.

References

Buschmann, C., Pfisterer, D., Fischer, S., Fekete, S., & Kröller, A. (2005). SpyGlass: a wireless sensor network visualizer. ACM SIGBED Review, 2(1): 1-6.

Calderón-Córdova, C., Bustamante, B. Delgado, J., Febres, C., Montano, V., Saritama, C., & Ramirez, C. (2018). Wireless sensor network for real-time monitoring of temperature, humidity and illuminance in an orchid greenhouse. In Proceedings of the 13th Iberian Conference on Information Systems and Technologies, Caceres, Spain, June 13-16.

Castillo, J., Ortiz, M. A., López, V., Olivares, T., & Orozco-Barbosa, L. (2008). WiseObserver: A Real Experience with Wireless Sensor Networks. In Proceedings of the 3rd ACM workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks, 23-26.

Chauhan, J., & Bojewar, S. (2016). Sensor networks based healthcare monitoring system. In Proceedings of the International Conference on Inventive Computation Technologies, 2:1-6.

d'Auriol, J. B., Lee, S., & Lee, Y. K. (2010). Visualizations of Wireless Sensor Network Data. Handbook of Research on Developments and Trends in Wireless Sensor Networks: From Principle to Practice. In H. Jin and W Jiang (Eds.), (pp. 353-370). IGI Global Publisher.

Davcev, D., Mitreski, K., Trajkovic, S., Nikolovski, V., & Koteli, N. (2018). IoT agriculture system based on LoRaWAN. In Proceedings of the 14th IEEE International Workshop on Factory Communication Systems, Imperia, Italy, June 13-15.

Dayarathna, M. (2019). Comparing 11 IoT Development Platforms. IoT Zone, [Online] Available: https://dzone.com/articles/iot-software-platform-comparison (July 12, 2019).

Djenouri, D., Laidi, R., Djenouri, Y., & Balasingham, I. (2019). Machine Learning for Smart Building Applications: Review and Taxonomy. Journal of ACM Computing Surveys, 52(2): pp1-42.

Elhakim, R., & Elhelw, M. (2010). Interactive 3d visualization for wireless sensor networks. The Visual Computer, 26(6-8):1071-1077.

Faludi, R. (2010). Building wireless sensor networks: with ZigBee, XBee, Arduino, and Processing. O'Reilly Media.

Alcala, H. A. G., Rivero-Angeles, E. M., & Anaya A. E. (2019). Primary User Emulation in Cognitive Radio-Enabled WSNs for Structural Health Monitoring: Modeling and Attack Detection. Journal of Sensors, 2019: 1-14.

Ghayvat, H., & Mukhopadhyay, S., C., Gui, X. & Suryadevara, N. (2015). WSN- and IOT-Based Smart Homes and Their Extension to Smart Buildings. Journal of Sensors, 15: 10350-10379.

Granda-Cantuna, J., Molina, C., Hidalgo-Lupera, S., & Valarezo-Varela, C. (2018). Design and Implementation of a Wireless Sensor Network for Precision Agriculture Operating in API Mode. pp144-149. In Proceedings of the International Conference on eDemocracy & eGovernment, Ambato, Ecuador, April 4-6.

Grosso, M., Rinaudo, S., Patti, E., & Acquaviva, A. (2018). An energy-autonomous wireless sensor network development platform. In Proceedings of the 13th International Conference on Design & Technology of Integrated Systems, Taormina, Italy, April 9-12.

Jurdak, R., Ruzzelli, A., Alessio, B., & Boivineau, S. (2011). Octopus: Monitoring, visualization, and control of sensor networks. Wireless Communications & Mobile Computing, 11(8): 1073-1091.

Liang, Y., Meng, X., Hu, Yi., & Zhang, K. (2017). Design and implementation of an ultra-low power wireless sensor network for indoor environment monitoring. In Proceedings of the 17th IEEE International Conference on Communication Technology, 937-940.

Liu, X., Chen, H., Wang, M., & Chen, S. (2012). An XBee-Pro based energy monitoring system. In Proceedings of the Australasian Telecommunication Networks and Applications Conference, Brisbane, QLD, Australia, November 7-9.

Ma, Y., & Chen, J. (2018). Toward intelligent agriculture service platform with lora-based wireless sensor network. In Proceedings of the IEEE International Conference on Applied System Invention, Chiba, Japan, April 13-17.

Medeiros, T., Carvalho, F., Souza, C., Henrique, P., & Molina, Y. (2018). Vegetation Encroachment Monitoring System for Transmission Lines Using Wireless Sensor Networks. In Proceedings of the IEEE International Instrumentation and Measurement Technology Conference, Houston, TX, USA, May 14-17.

Minoli, D., Sohraby, K., & Occhiogrosso, B. (2017). IoT Considerations, Requirements, and Architectures for Smart Buildings – Energy Optimization and Next Generation Building Management Systems. IEEE Journal of Internet of Things, 4(1): 269-283.

Mitton, N., Papavassiliou, S., Puliafito, A., & Trivedi, K. (2012). Combining Cloud and sensors in a smart city environment. EURASIP Journal on Wireless Communications and Networking, 2012: 247.

Monteiro, L. P., Zanin, M., Menasalvas, R. E., Pimentao, J., & Sousa, A. P. (2018). Indoor Temperature Prediction in an IoT Scenario. Sensors (Basel, Switzerland) 1,8(11): 3610.

Nagaraj, S & Rajashree, V. B. (2017). Applications of wireless sensor networks in the real-time ambient air pollution monitoring and air quality in metropolitan cities — A survey. In Proceedings of the International Conference on Smart Technologies for Smart Nation, 1393-1398.

Nooriman, W. M., Abdullah A. H., Rahim N. A., & Kamarudin, K. (2018). Development of wireless sensor network for Harumanis Mango orchard's temperature, humidity and soil moisture monitoring. In Proceedings of the IEEE Symposium on Computer Applications & Industrial Electronics, 263-268.

Onibonoje, M., & Olowu, T. (2018). Real-time remote monitoring and automated control of granary environmental factors using wireless sensor network. In Proceedings of the IEEE International Conference on Power, Control, Signals and Instrumentation Engineering 113-118.

Othman, B. S., Trad, A., & Youssef, H. (2014). Security architecture for at-home medical care using Wireless Sensor Network. In Proceedings of the International Wireless Communications and Mobile Computing Conference, 304-309.

Parbat, B., Dwivedi, A. K., & Vyas, O. P. (2010). Data Visualization Tools for WSNs: A Glimpse. International Journal of Computer Applications, 2(1): 14-20.

Ravichandran, S., Chandrasekar, R., Uluagac, S., & Beyah, R. (2016). A Simple Visualization and Programming Framework for Wireless Sensor Networks: PROVIZ. Ad Hoc Networks, 53: 1-16.

Ray, P. P. (2015). Internet of Things based smart measurement and monitoring of wood Equilibrium Moisture Content. In Proceedings of the 2015 International Conference on Smart Sensors and Systems, Bangalore, India.

Ray, P. P. (2016). An Internet of Things based approach to thermal comfort measurement and monitoring." In Proceedings of the 3rd International Conference on Advanced Computing and Communication Systems, 465-471.

Shu, L., Wu, C., Zhang, Y., Chen, J., Wang, L., & Hauswirth, M. (2008). NetTopo: Beyond Simulator and Visualizer for Wireless Sensor Networks. In Proceedings of the IEEE Second International Conference on Future Generation Communication and Networking, 17-20.

Sriploy, P. & Chanpuek, T. (2018). The enhancement of wireless sensor network in smart farming using distributed beamforming. In Proceedings of the International ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering, Chiang Rai, Thailand, Feb. 25-28.

Sunehra, D. & Rajasri, S. (2017). Automatic street light control system using wireless sensor networks. In Proceedings of the IEEE International Conference on Power, Control, Signals and Instrumentation Engineering, pp.2915-2919.

Syafrudin, M. Alfian, G. Fitriyani, N. & Rhee, J. (2018). Performance Analysis of IoT-Based Sensor, Big Data Processing, and Machine Learning Model for Real-Time Monitoring System in Automotive Manufacturing. Sensors, 18(9): 2946.

Turon, M. (2005). MOTE-VIEW: A sensor network monitoring and management tool. In Proceedings of the Second IEEE Workshop on Embedded Networked Sensors, 11-18.

Wei, X. Geng, L. & Zhang, X. (2017). An Open Source Data Visualization System for Wireless Sensor Network. Journal of Computer Science and Information Technology, 5(2): 10-17.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of Things for smart cities. IEEE Internet of Things, 1: 22-32.