Design and Evaluation of an Open-Source Wireless Mesh Networking Module for Environmental Monitoring

Huang-Chen Lee, *Senior Member*, IEEE, Hsiao-Hsien Lin

Abstract—Wireless mesh networking extends the communication range among cooperating multiple low-power wireless radio transceivers and is useful for collecting data from sensors widely distributed over a large area. By integrating an off-the-shelf wireless design, such as the XBee module, development of sensor systems with mesh networking capability can be accelerated. This study introduces an open-source wireless mesh network (WMN) module, which integrates the functions of network discovery, automatic routing control, and transmission scheduling. In addition, this design is open-source in order to promote the use of wireless mesh networking for environmental monitoring applications. Testing of the design and the proposed networking module is reported. The proposed wireless mesh networking module was evaluated and compared to XBee. The average package delivery ratio and standard deviation of the proposed WMN module and the XBee are 94.09%, 91.19%, 5.14% and 10.25%, respectively, in a 20 node experiment. The proposed system was demonstrated to have the advantages of low-cost combined with high reliability and performance, and can aid scientists in implementing monitoring applications without the complications of complex wireless networking issues.

Index Terms—XBee, wireless sensor, mesh network module, open-source, 802.15.4, ZigBee

I. Introduction

Using a wireless network to monitor the environment has the advantage of imposing no constraints due to cabling for data transmission. Wireless mesh networking combines multiple wireless transceivers to cooperatively transmit and relay their data, thereby extending the region of communication. Wireless sensor networks (WSNs) based on this characteristic enable environmental monitoring of large areas with multiple wireless sensors. Environmental scientists already take advantage of WSNs to facilitate the exploration of our world.

WSNs have been used in many applications, including health care [1], home monitoring [2], green house management [3], power consumption monitoring [4][5], agricultural irrigation control [6] and snail pest detection [7]. Wireless mesh networking of WSNs is the key feature making these applications possible. The ZigBee protocol [8] is one of the most popular designs that supports wireless mesh networking based on the IEEE 802.15.4 standard [9]; therefore, environmental scientists can integrate an off-the-shelf ZigBee module, i.e., the XBee design [10], into their sensor system and enable wireless mesh networking with minimal effort.

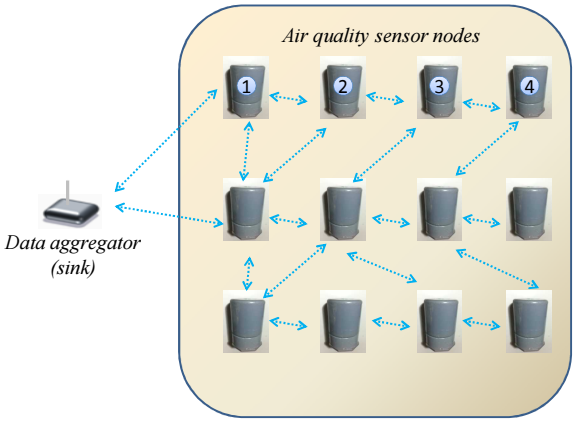


Fig. 1. An example of wireless mesh network for air quality monitoring.

For example, environmental scientists can integrate air quality sensors with the XBee module and deploy these wireless sensors in a distributive manner to measure atmospheric pollutant concentrations in different locations of the same region. The study [11] presents a typical scenario that using a wireless mesh network for air quality monitoring. As shown in Fig. 1, The sensors may be deployed in a square-grid topology with equal distances, and each sensor cover the same size of sensing region. The collected data of sensors send back to a data server through a wireless mesh network for later analysis and validation. Wireless mesh network can help to deliver data of sensors even if some sensor nodes cannot directly communicate wirelessly with the data aggregator (or sink), by requiring other nodes to relay the data and transfer it to the sink. Given the example scenario in Fig. 1, the sensor 4 cannot deliver its data directly to the sink, and sensor 4 asked the other sensors, including sensor 3, 2 and 1 to help relaying its data to the sink. Wireless mesh network can extend the region of communication for helping the geographical distributed sensors to deliver data in an efficient manner. Other potential applications like collecting human data with wearable sensors [12], detecting pipe leakage with pressure sensors [13], can be enhanced by using wireless mesh network to extend the region of monitoring.

However, the use of such commercial products raises issues that prohibit applications of this kind of wireless module. One major concern is that the hardware and source code are not made available to the general public because the commercial product is based on proprietary software. During the design

stage of a sensor system, users may neither understand how the network of the module was formed, nor how the routing topology was created. Without access to the source code, or a detailed design of the wireless network module, the root cause of a problem is difficult to determine. In order to respond to this issue, a wireless mesh networking (WMN) module is proposed in this study (shown in Fig. 2) that can be easily integrated into the sensor system. The source code of this design is open to the public to promote the usage of mesh networking.

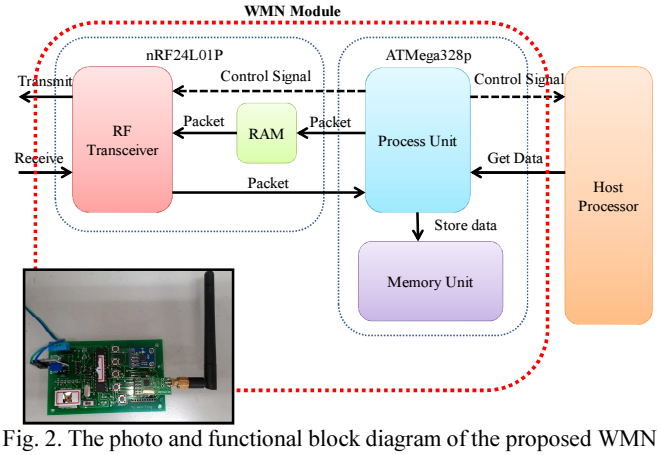


Fig. 2. The photo and functional block diagram of the proposed WMN module.

This study is extended from our previous work [14]. The goals of developing a WMN module for WSN applications include: (1) forming a wireless mesh networking topology, (2) adjusting the topology according to the changes of the radio interference, environment, obstacles, etc., (3) transferring the sensor's data back to the sink, gateway, or data aggregator (usually a computer, i.e., a PC), and (4) subsequently uploading the aggregate data to the remote server.

These goals roughly depict the functions that a WMN module must demonstrate. However, the various types of WSN applications may have different requirements for data transmission. For example, the measurement of air quality may generate 10 bytes of data per 10 minutes per node. Conversely, a WSN for monitoring streetlight lamps may need to measure current, voltage, temperature, and lighting intensity of a lamp at one Hz, which generates up to 30 bytes of data per second per node. A survey of existing WSN applications [10] reveals that the requirements are diverse. Nevertheless, it is essential for a WMN module to provide reliable transmission at a relatively low data rate , which may satisfy most WSN applications and was the initial goal for our design. In summary, the significant contributions of this study are:

1. A WMN module is proposed, which can be easily integrated with a sensor system to enhance it by extending the communication range and sensing coverage without the hassle of a network configuration.
2. The module design is based on an 8-bit microprocessor and a simple radio frequency (RF) transceiver without special functional support (i.e., Radio Signal Strength Indicator (RSSI), which is not supported by some inexpensive RF transceivers, i.e., Nordic nRF24L01P). Therefore, this design can be ported to other microprocessors and RF transceivers.
3. The proposed wireless module is open-source in both software and hardware. Therefore, it can not only be integrated into a sensor system for environmental monitoring, but it can also be used to study the performance of wireless mesh networking in an actual experiment, and to modify it depending on different requirements. Therefore, the entry-level knowledge required by users of wireless mesh networking is reduced.

The remainder of this paper is organized as follows: Section II discusses related works and the goals of this project; Section III describes the details of the architecture and design of the system; Section IV illustrates the evaluation of the proposed system; and Section V presents the conclusions of this study.

II. Related Works and Design Goals

In this section, research related to the WMN module is discussed. As shown in Fig. 3, a WMN module is connected to the host processor via a GPIO, UART, SPI or I2C interface. The host processor reads data from its sensors (i.e., temperature, humidity, or air quality, etc.), and sends the collected data to the WMN module. The WMN module buffers the data and transfers it to the destination node in a wireless mesh network.

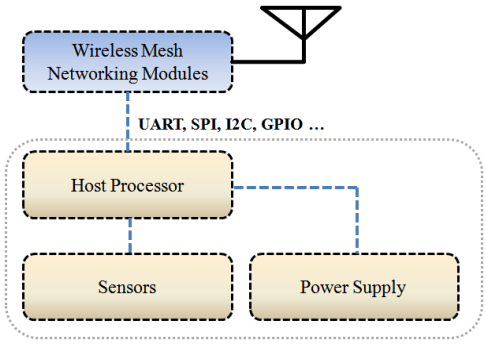


Fig. 3. Architecture of a typical sensor node integrated with a WMN module.

XBee [10] is a popular WMN module for integration with sensor systems. Variant versions of the XBee WMN module support different protocols and standards, including 802.15.4 [9], ZigBee [8], and the proprietary DigiMesh [15]. Many monitoring applications [1]-[7] are based on the XBee WMN module. Similar off-the-shelf modules [21][22][23] are also available to be used as a WMN interface in a sensor system.

In addition to using a low-power RF transceiver, like IEEE 802.15.4/TI CC2530 or Nordic nRF24L01P, many Wi-Fi based wireless mesh network projects [24][25] have been studied previously for providing a larger coverage area and a decentralized network infrastructure, which is suitable for connecting computers in rural areas or preventing networks having to be controlled by a centralized authority. These projects are implemented on a PC or Wi-Fi network router; therefore, both the hardware cost and power consumption are higher than the proposed approach. These issues make using Wi-Fi to form a wireless mesh network for environmental monitoring infeasible in many practical applications.

In the following section, we discuss common issues with using such WMN modules in real applications. In the deployment stage of a WMN module, it is common to experience the problem that some nodes cannot transmit data reliably to the sink. However, it is difficult to know the root cause as no simple method exists to trace the routing path of the data packet in the ZigBee or DigiMesh modules. The problem is that the decisions for routing paths in the ZigBee and DigiMesh modules are based on Ad hoc On-Demand Distance Vector routing (AODV), which explores the routing path while a node is attempting to send data. The benefit of AODV is that the node in the mesh network is not required to maintain a memory-costly large routing table.

Because the ZigBee is based on a comprehensive standard, including forming the network, addressing the nodes, data security, profiles and other features. The MeshBee system [18] is an open-source ZigBee module based on the Jennic JN5168 [19]—a wireless microcontroller with an 802.15.4 RF transceiver. The open-source format aids the user in debugging and customizing their system. Nevertheless, the ZigBee implementation is comprehensive, it is difficult to understand and modify without a steep learning curve. Another open source project [20] by Swiftlet Technology had been announced with a similar goal as ours, but the recent progress is not available as the funding was unsuccessful through KickStarter [20].

In consideration of these issues, the following design goals were considered in this study:

1. The proposed open-source code of the WMN module should be compact in contrast to the more complex ZigBee design to allow the user to use and test the system without the delay of a steep learning curve.
2. The correctness and reliability of the proposed WMN module must be verified by application in actual experiments to demonstrate that its performance is comparable to an off-the-shelf module.
3. The design may not limited by the specific hardware platform, which should reduce the costs of implementation.

These design requirements are summarized in the design specifications of the WMN module and are described in the following section.

III. Design of WMN Module

In this section, the design of the WMN module is described.

A. Architecture of WMN

The proposed WMN module (Fig. 2) is based on a generic 8-bit microprocessor (Atmel ATMega328p) and a low-cost RF transceiver (Nordic nRF24L01P), whose output power is 1 mW at 2.4 GHz. Interfacing to the host processor is accomplished via implementation of UART and GPIO. A functional block diagram of a WMN module is presented in Fig. 2. The primary processor of the WMN module (ATMega328p) is connected to the host processor for interfacing the control signal and the data transferred from the host processor. The nRF24L01P is connected to and managed by the ATMega328p microprocessor via the SPI/I2C. In this design, the host processor only communicates with the ATMega328p microprocessor when sending and receiving the data packet. The host processor does not need to be concerned with the transition state among multiple modes, as shown in Fig. 4; therefore, the host processor can focus on the application level design and utilize the WMN module as a radio modem.

B. Operating Modes of a WMN Module and Detailed Flowcharts

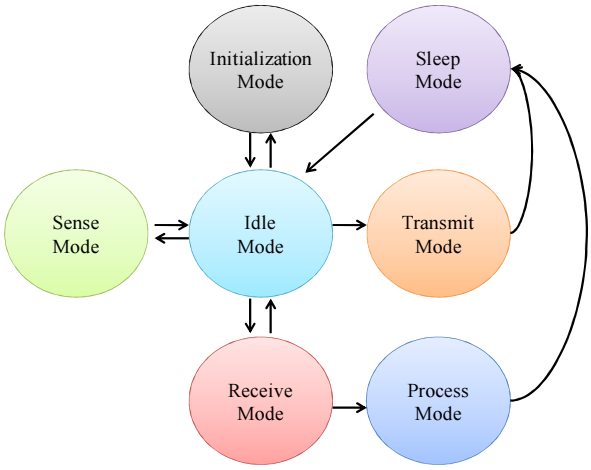


Fig. 4. Finite state machine of a WMN module.

The finite state machine (FSM) of a WMN module (connected to a sensor node) is illustrated in Fig. 4. As the module is powered on, the WMN module starts in the Initialization Mode while waiting to join a network. The system switches to the Idle Mode after joining a network. Subsequently, the WMN module switches its running mode according to the work schedule assigned from the sink, including *Sense Mode*, *Sleep Mode*, *Process Mode*, *Transmit Mode*, and *Receive Mode*. The RF transceiver only turns on in Transmit Mode and Receive Mode to save energy. The purpose of Sense Mode is to read data from the analog-digital converter (ADC) of the microprocessor of the WMN module, this design provides a similar function as XBee, it allows for reading sensor data from an external sensing component (i.e., a temperature sensor) without support from a host processor.

The proposed WMN module is based on Time Division Multiple Access (TDMA) protocol and is sensitive to timing control. The module needs to switch to suitable modes as listed in Fig. 4 using exact timing. Because no operating system is required in the design, it uses a hardware timer interrupt of the microcontroller to trigger the timer interrupt service routine (ISR) for ensuring timing correctness. As the timer ISR was triggered in the beginning of a new timeslot, as shown in Fig. 5, it toggles the Boolean variables corresponding to different modes, i.e., Process Mode, Receive Mode, Transmit Mode, Sensing Mode, etc. The timer ISR only modifies the Boolean variables and ends the ISR immediately. The variables will be referred to in the main loop program to control the system behaviors.

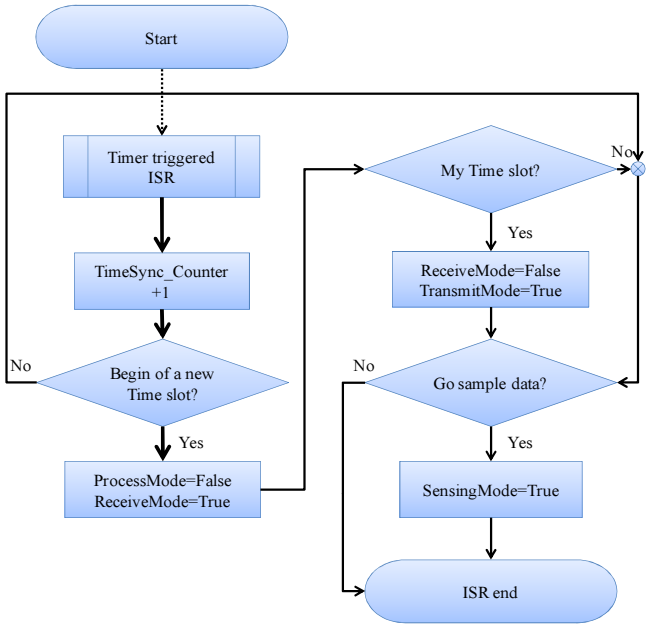


Fig. 5. Flow Chart: Timer ISR driven changes of WMN module's mode.

B.1 Initialization Mode

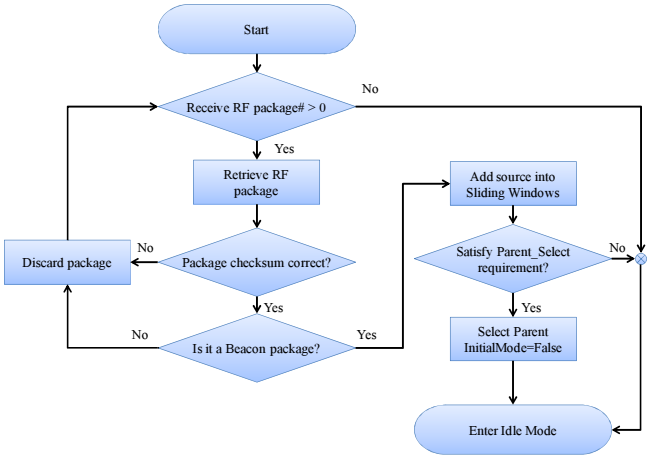


Fig. 6. Flow Chart: Initialization mode.

In Initialization Mode, as shown in Fig. 6, the microcontroller of the WMN module checks whether it received any packages from the RF transceiver, and validates the checksum correctness. If the package is valid and it is a Beacon (from another router), it adds the source node ID into the parent buffer for later use by the Parent_Select function (which will be introduced later in this section). If the beacon queue buffers sufficient node IDs, then it executes the Parent_Select function to determine the best parent candidate; the module joins the selected parent's network and enters Idle Mode.

B.2 Idle Mode

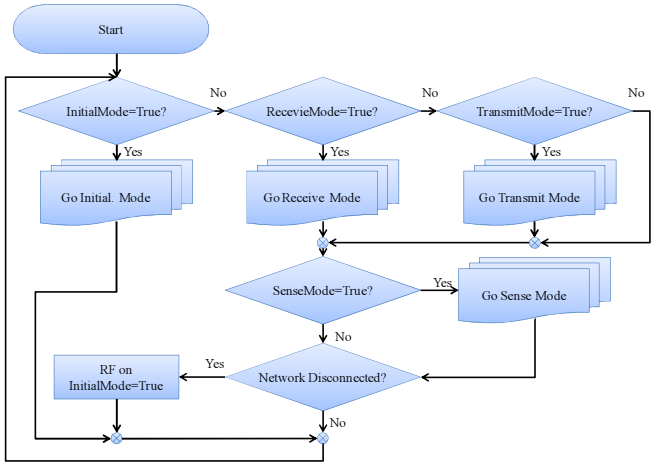


Fig. 7. Flow Chart: Idle mode.

Depicted in Fig. 7, the microcontroller of the WMN module in Idle Mode periodically checks the Boolean variables to go to different modes, which were modified by the timer ISR. The Initial Mode, Receive Mode and Transmit Mode are mutually exclusive. The Sense Mode is checked for whether it needs to sample data from the sensing component connected to the WMN module. At the end of Transmit Mode and Receive Mode, it also checks the status of the network connection (i.e., is the current parent reachable?), and will execute Initial Mode to find another, better parent if necessary.

While considering low speed microcontrollers and potential buffer-overrun of the RF transceiver, the receiving procedure is separated into two modes, Receive Mode and Process Mode, to deal with incoming data in a timely manner. The procedure involves retrieving data from the buffer of the RF transceiver, validating the incoming data package using a checksum algorithm, determining the package type, synchronizing its clock to the beacon , data conversion, and so on. In these steps, data conversion, storage (into flash memory or external storage, i.e., SD card) and display messaging (via UART for debugging) are time consuming and often introduce un-deterministic delay and buffer-overrun of the RF transceiver. As illustrated in Fig. 4, Receive Mode retrieves the data package and validates the checksum, and Process Mode deals with data conversion, storage and display. This avoids a single function/mode dominating the microprocessor for too long.

B.3 Receive Mode and Process Mode

Receive Mode (shown in Fig. 9), starts by recording the current start time of the timeslot, then turns on the RF transceiver and waits for an incoming data package. This mode ends after timeout. If a package is received, it checks the checksum and forwards the data package to the corresponding path. In this mode, we synchronize the internal clock if it is a valid beacon package from its parent.

In order to save energy, Process Mode (Fig. 10) checks the corresponding variable and if it is false, goes to Sleep Mode immediately. Otherwise, it processes data in the buffer until the buffer is empty. As this design is timing sensitive, the total time used by Receive Mode and Process Mode must not exceed the length of a timeslot. In this design, the timer ISR toggles Process

Mode to false if the runtime exceeds the timeslot, therefore, ensuring the timing correctness between several nodes over timeslots.

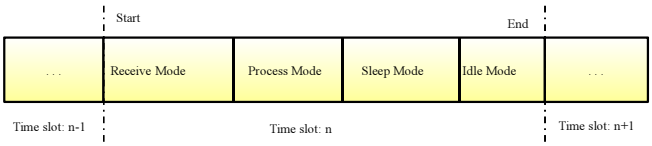


Fig. 8. A detail schedule in a timeslot for receiving data.

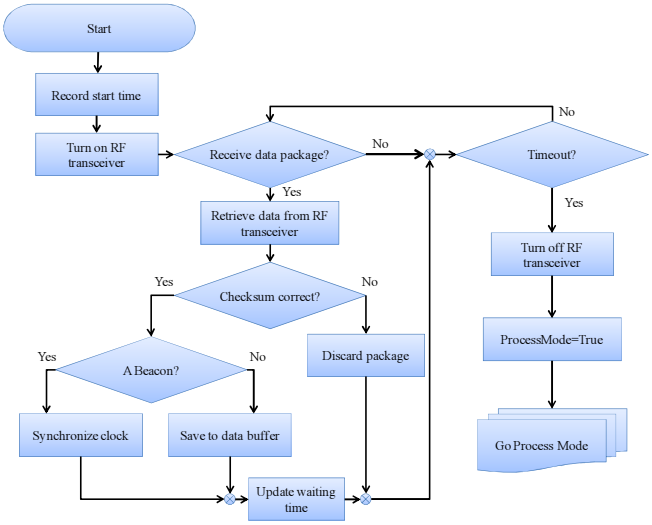


Fig. 9. Flow Chart: Receive mode.

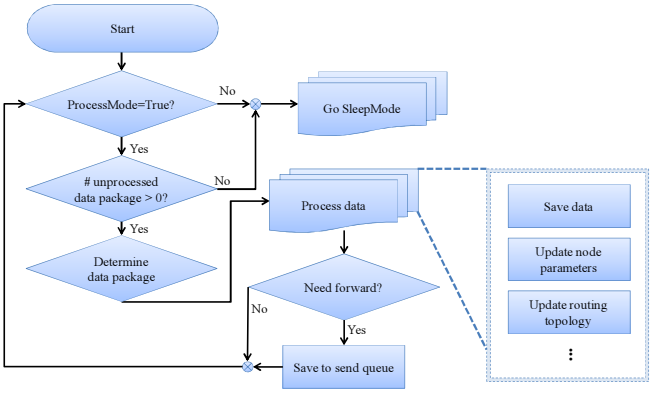


Fig. 10. Flow Chart: Process mode.

B.4 Sense Mode

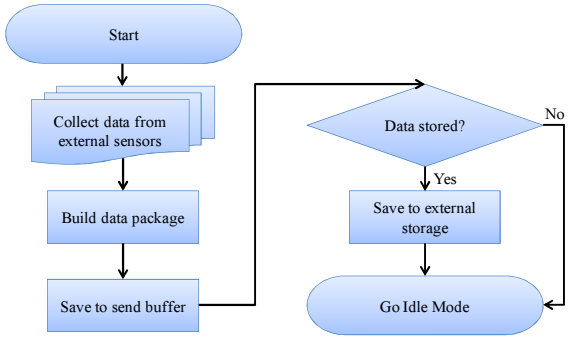


Fig. 11. Flow Chart: Sense mode.

In Sense Mode, the microprocessor collects data from its external sensors (like the ADC implementation in XBee), because the proposed WMN module can read from external sensors independently without support from a host processor.

B.5 Transmit Mode

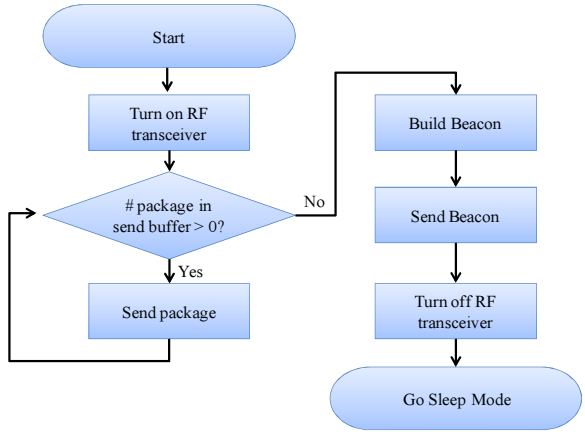


Fig. 12. Flow Chart: Transmit mode.

On the WMN module's timeslot, it sends all the data in the buffer to the RF transceiver. After the send buffer is empty, it goes to Sleep Mode to save energy.

C. Formation of Mesh Network: A TDMA approach

The design of the proposed WMN module is aimed at forming a wireless mesh network automatically. Three types of roles are defined in design of the mesh network: *End Device* (ED), *Router* and *Sink*. One of the modules in the network is set as the Sink to collect data from all other modules. The other modules in the initial state are acting as an ED. An ED is a simple wireless sensor, collecting data and sending it back to its parent. An ED changes its role to a Router if the ED can help another ED/Router to relay data to its parent.

In contrast to ZigBee, which is based on carrier sense multiple access with collision avoidance (CSMA/CA) and Ad hoc On-Demand Distance Vector Routing (AODV), the proposed design uses Time Division Multiple Access (TDMA) while keeping design complexity and manageability in mind. This is further strengthened by the goals of most WSN applications that are designed to collect data from a WSN. Data is often periodically generated from the sensor nodes and is collected and transferred to the sink. TDMA is suitable in such applications as the TDMA schedule reserves a timeslot for each node in the system, and increases the package delivery ratio and reduces package collision in wireless media.

In the design, each node in the system was pre-assigned a unique ID number and has a corresponding and exclusive timeslot in every TDMA cycle, to avoid multiple WMN sensor nodes trying to transmit data at the same timeslot. In this design, the WMN module with the ID 0 is functioning as a sink, which is used to:

(1) Broadcast the initial beacon signal and TDMA schedule to manage other nodes
(2) Collect the data from nodes

The other nodes (which the ID is not 0) are sensor nodes (or ED/Router), which sense data and transfer it to the sink. Fig. 13 presents the working flowchart of a WMN sensor node. At the beginning of a timeslot, WMN checks whether the current timeslot belongs to itself by checking whether WMN's ID = current timeslot number.

(1) If WMN's ID＝current timeslot number
If the WMN has joined a network, then it broadcasts its beacon to notify its near neighbors. If it has not yet joined any network, it skips this round.
(2) If WMN's ID ≠ current timeslot number
If a node does not have the current timeslot and receives a beacon from another node, then it will run the Parent_Select function, if it has not yet joined any network.

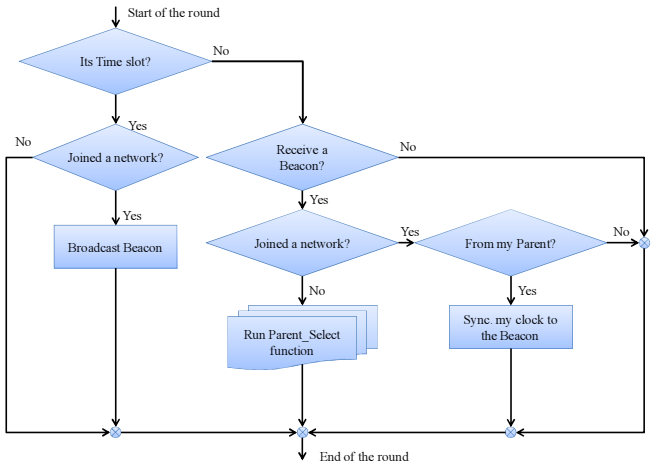


Fig. 13. Flow Chart: Mesh network formation from the view of a WMN module

D. Parent_Select Function: A Hardware-Independent Approach

The WMN modules form a mesh network and route data based on a *hop-count distance* estimation approach, rather than using Radio Signal Strength (RSS) or LQI (Link Quality Indicator) to determine the quality of a path. Because RSS and LQI are not always available, i.e., Nordic nRF24L01P is not capable of measuring RSSI and LQI. Therefore, in this design the parent node of the WMN module is determined by selecting the node with the smallest hop count.

Meanwhile, the parent node will be changed if:
(1) The current parent is not available (not receiving its beacon) for a predefined length of time, or
(2) A better parent becomes available, even if its hop count is equal to the hop count of the current parent

The received beacons from parent candidates are stored in a FIFO queue. If a WMN module receives beacons from multiple parents it determines the best candidate according to the frequency of the beacons received within the sliding window of the beacon queue. This approach has the advantages that the computation complexity is fixed and the memory requirement is small. Since in a network, there might be some nodes is unreliable, due to hardware failure or network problem, but still appeared to be a good parent node candidate to others. It might cause PDR decrease and significant transmission delay if such a node been selected as a parent node. Therefore, in this design, each node has a blacklist that is used to mark such an unhealthy parent node, and prevent the node from being selected repetitively.

IV. Evaluation and Discussion

The proposed WMN module was evaluated to verify the correctness of the design. Twenty wireless sensor nodes based on the proposed WMN module were deployed over an office floor at the university campus, while twenty XBee DigiMesh 2.4 based wireless sensors were also installed in the same locations (Fig. 14) for performance comparison. Table I presents the details of the two versions of wireless sensor node. Note that for the sensor with XBee, an additional microcontroller Atmel Atmega328p is connected to the XBee (Fig. 14 right) to control the timing of sample and sending data, therefore, synchronizing the frequency of data generation from the two types of nodes.

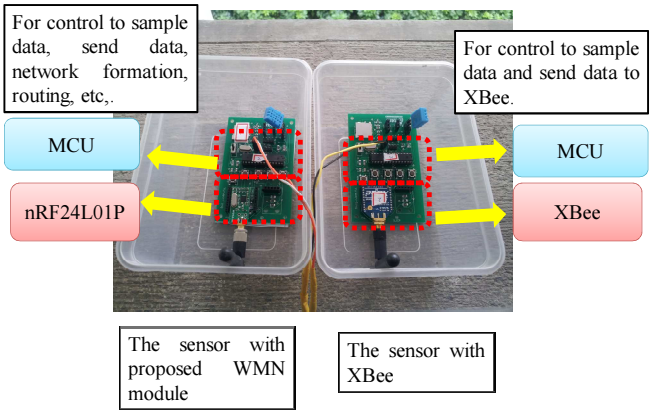


Fig. 14. The proposed WMN module (left) and off-the-shelf XBee DigiMesh 2.4 (right).

The WMN module was configured at an output level of 1 *mW* RF at 2.4 GHz with a 3dBi omni-directional antenna, the same as the XBee DigiMesh 2.4. We set timeslots to 1 second and a TDMA cycle contains 20 timeslots in our WMN module. Each node was configured to send one data packet per minute to simulate a periodic reporting application. The data packet will eventually be relayed by the other nodes and delivered to the sink (denoted as 'S' in the Fig. 18). The experiment was conducted for five days to evaluate the performance of the packet delivery and the stability of the network topologies. Table I shows the configuration details.

In this experiment design, some sensor nodes (at location 16, 17 and so on) were intentionally installed in the open space at the left-side shown in Fig. 18, where there is no Line-of-Sight

(LOS) to the sink. Therefore, these sensor nodes always need other nodes for relaying data to the sink. In addition, we can evaluate how the higher deployment density and number of sensor nodes can affect the network performance. This experiment was conducted for 218 hours or about 9 days.

Table I: Configurations of the sensor with WMN module and XBee

	WMN Module (in Fig. 14 left)	**XBee DigiMesh** (in Fig. 14 right)
Protocol	The proposed protocol	DigiMesh
MCU	ATmega328P, 16MHz (for control sending data, network formation, routing, etc,.)	ATmega328P, 16MHz (for control sending data)
RF Transceiver	Nordic nRF24L01P	XBee DigiMesh
RF Frequency	2.4GHz	2.4GHz
RF Power	1mW	1mW
Antenna	Omni Directional SMA, 3dBi	Omni Directional RP-SMA, 3dBi

1) Dynamic Routing

As shown in Fig. 18(a), we can see that in the beginning of the experiment all the WMN sensor nodes can reach the sink. The WMN sensor at location 16 can transfer its data along the path 16-19-1-S(sink). After several days, the topology changed as shown in the snapshot in Fig. 18(b). In addition, the WMN module at location 13 (hereafter denoted by *WMN 13*) stopped working due to hardware failure. This can be expected based on Fig. 18, where none of the WMN nodes selected WMN 13 as their parent, because it did not seem to be a reliable parent candidate. The performance of dynamic routing is presented in the figure. The WMN module at location 16 in Fig. 18(a) transfers its data along the path 16-19-1-S, but changed the path to 16-17-14-S. This indicates that WMN 16 considered WMN 17 a better parent in Fig. 18(b), based on the Parent_Select function.

2) Package Delivery Ratio and Received Package Number

We further investigated the package delivery ratio and received package number during the experiment, for both the proposed WMN module and the XBee as present in Fig. 15. This figure shows the received number of data packages from the WMN module and the XBee are all very similar in most locations; except the WMN at location 13 and the XBee at location 9, which malfunctioned due to hardware failure, therefore, the received package numbers are much less. The power outage of the sensors at location 11 during the experiment also caused a fewer number of data packages received.

For the package delivery ratio (PDR, the percentage of data sent from the sender that can be delivered to the sink), the proposed WMN module-based wireless sensors are relatively stable among various locations, as the PDRs of XBee have higher fluctuations. As shown in Fig. 15, the average PDR and standard deviation of the proposed WMN module and XBee are 94.09%, 91.19%, 5.14% and 10.25%, respectively. Under the same configurations, the experiment results support the proposed WMN module can offer comparable or even better performance than commercial products such as the XBee, since the XBee's standard deviation of PDR is higher than the proposed WMN showing it is relative unreliable.

3) PDR vs. Number of Parent Change and Average Hops

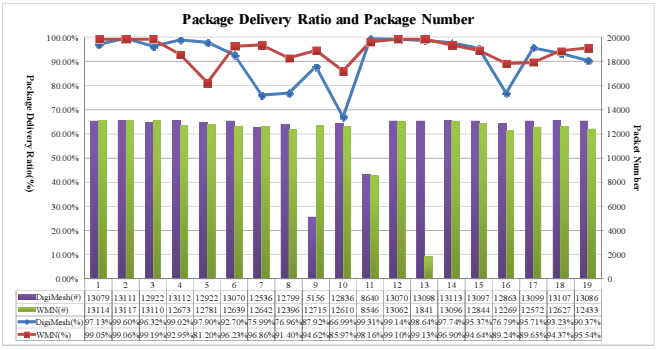


Fig. 15. Package delivery ratio and received package number.

In the design of the proposed WMN module, the change parent is triggered if (1) the current parent disappears, or (2) a better parent candidate is present, meaning we may infer the link between a node and its current parent is unstable. Therefore, the parent change is the result of detecting a weak link, but not the root cause of a low PDR. A weak link may occur due to (1) the distance between nodes being too far, (2) obstacle appear, or (3) existence of co-channel interference. Because we were unable to retrieve any networking information from the XBee, therefore, analysis of the root cause of package loss of XBee is not present in this study.

Fig. 16 shows the relationship between PDR and Number of Parent Change (hereafter denoted by NPC). We can observe that WMN 16 has a very high NPC, approximately 920, but the PDR is not the lowest. The lowest PDR in this experiment is on WMN 5, for which the NPC is only about 140. The correlation coefficient between PDR and NPC is -0.41, which indicates the relationship is not significant.

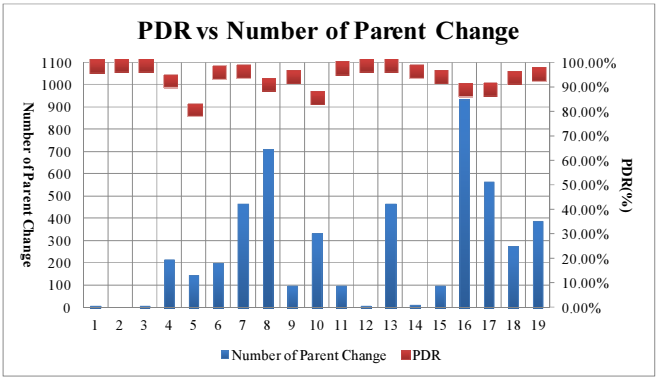


Fig. 16. PDR vs Number of Parent Change

Fig. 17 investigates how the average hop(s) of a node's transfer of the data to the sink may affect PDR. WMN 16 holds the highest average hops, approximately 2.8, due to its location

does not line-of-sight to the sink. WMN 16 must rely on other nodes, such as WMN 17, 18 and 19, to help relay its data delivery to the sink. For the locations close to the sink, i.e., WMN 1, 2, 3, 11, 12, the average hop(s) is 1 as they do not need other nodes to relay data, they can reach the sink directly. The correlation coefficient between PDR and the average hop(s) is -0.60, which indicates a moderate negative relationship.

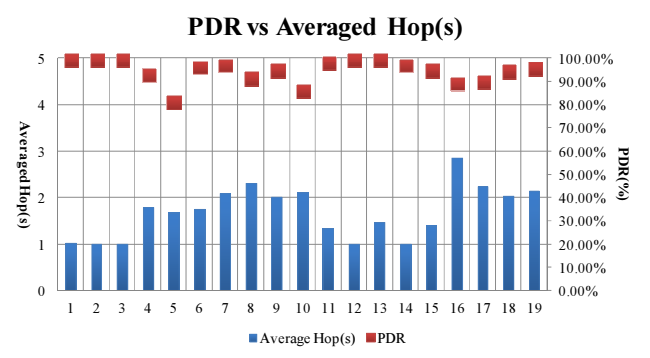


Fig. 17. PDR vs Averaged Hop(s)

Table II: The data paths of WMN 16.

Path	Package Number	Percentage
16-14-S	780	7.12%
16-15-S	1577	14.40%
2 Hops	**2357**	**21.53%**
16-15-1-S	1859	16.98%
16-18-1-S	148	1.35%
16-19-1-S	894	8.17%
16-15-3-S	165	1.51%
16-19-3-S	73	0.67%
16-17-14-S	4227	38.61%
16-18-14-S	38	0.35%
3 Hops	**7404**	**67.62%**
16-17-18-1-S	351	3.21%
16-19-6-1-S	29	0.26%
4 Hops	**380**	**3.47%**
others	808	7.38%

We further examined the data path of WMN 16 as it is at the remotest location from the sink. Table II shows all the data paths of WMN 16. It shows WMN 16 changes its path whenever a better path is available, and the path 16-17-14-S (Sink) is chosen 38.61% of the time, more often than all other paths. We further investigated the PDR of WMN 16, 17 and 14 over the first 154 hours of the experiment, since these three nodes are usually connected and form a path to the sink, as shown in Fig. 18(b) and in Table II.

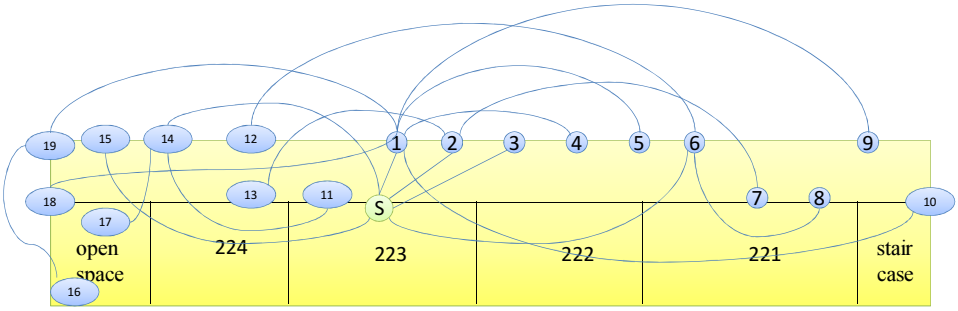
4) ***Reliability***

Fig. 19 shows the PDR of WMN 14 is relatively stable over time, because WMN 14 is close to the sink and can often connect to the sink with single-hop. In contrast, WMN 17 and 18 need other nodes to relay their data to the sink mostly; they usually take two or more hops to reach the sink. Therefore, the PDR of WMN 17 and 18 have greater fluctuation, which implies that more hops almost always cause instability. This may help us to improve the Parent_Change function to select a parent with fewer hop counts, although occasionally it might not be the best choice.

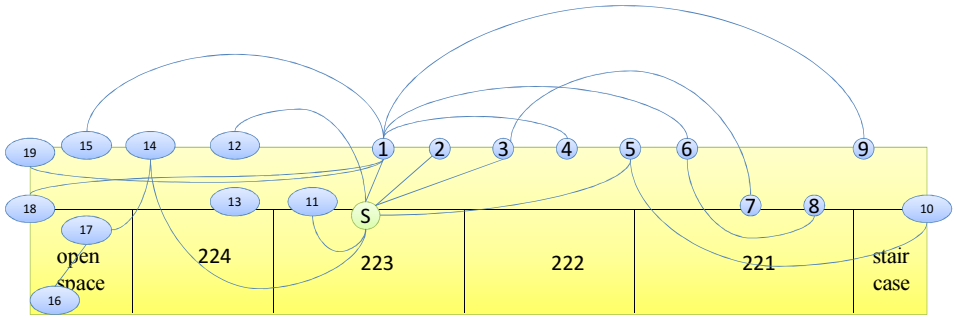
In the design, a node will automatically forward data from its child to its parent node in its timeslot, to carry out multi-hop data delivery. Many reliable transport protocols have been studied comprehensively [26]-[28] in the last decade, based on the feedback message from nodes to recover and resend the missing packages. The protocols can help to increase the completeness of data delivered to the sink. However, the implementation of these reliable transport protocols does increase the network message and design complexity, and will make the system difficult to handle. In addition, the WMN module needs to save all the transferred data in temporary storage, i.e., RAM or flash memory, which raises the memory requirement of the WMN module. This contradict to the original goals of this design that should made of an inexpensive, memory limited microprocessor. To strike to the balance between the design complexity and data reliability, the practical solutions to deal with this issue are to implement transmission-acknowledgement mechanisms in the application layer, or to send the same data redundantly to multiple parent nodes, in order to increase the delivery ratio of data that can eventually transfer to the sink.

V. Conclusions and Future Work

In this research, the design and implementation of a prototype WMN module were presented, and its performance in an actual experiment was evaluated. The proposed WMN module was evaluated and compared to XBee, an off-the-shelf product. The average PDR and standard deviation of the proposed WMN module and XBee are 94.09%, 91.19%, 5.14% and 10.25%, respectively, in a 20 node experiment. The results support that the proposed WMN module can offer comparable or even better performance than commercial products. In addition, it is a hardware-independent design because no special hardware capability is required. This design is available as open-source and can help to promote the use of wireless mesh networks for environmental monitoring. In the future, we envision further investigating the performance and power consumption of the proposed WMN module, and keep improving this design to aid scientists in implementing monitoring applications with less efforts on wireless networking issues.



(a) The snapshot at 21:43, 2014/06/4.



(b) The snapshot at 00:16, 2014/06/12.

Fig. 18. The deployment location and network topology snapshot of the WMN sensor nodes. The distance between location 10 and 18 was approximately 73 m.

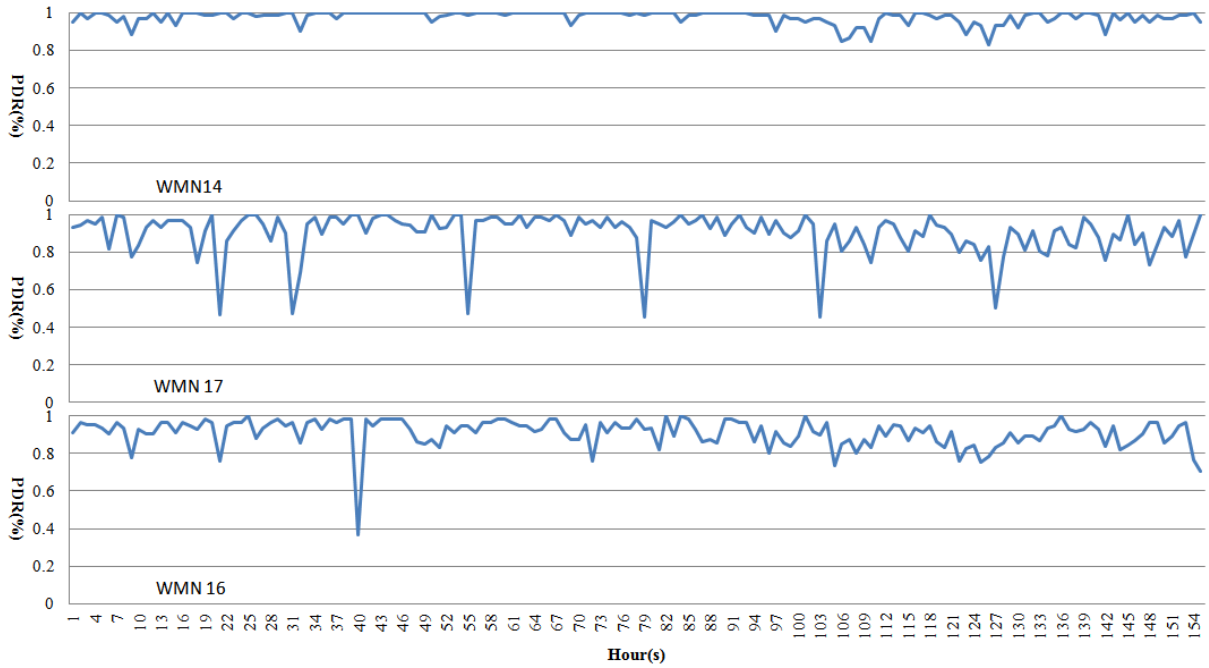


Fig. 19. The PDR of WMN 16, 17 and 14 over the first 154 hours of the experiment. The results implies the nodes with more hops to the sink have greater PDR fluctuation, and more hops almost always cause instability.

ACKNOWLEDGEMENTS

The authors would like to thank Yen-Shuo Huang, Wen-Yin Yu, Pei-Jyi Lee and Pin-Chen Kuo for their excellent technical assistance.

REFERENCES

[1] Suryadevara, N.K.; Mukhopadhyay, S.C., "Wireless Sensor Network Based Home Monitoring System for Wellness Determination of Elderly," Sensors Journal, IEEE , vol.12, no.6, pp.1965,1972, June 2012

[2] Kelly, S.D.T.; Suryadevara, N.K.; Mukhopadhyay, S.C., "Towards the Implementation of IoT for Environmental Condition Monitoring in Homes," Sensors Journal, IEEE , vol.13, no.10, pp.3846,3853, Oct. 2013

[3] Mirabella, O.; Brischetto, M., "A Hybrid Wired/Wireless Networking Infrastructure for Greenhouse Management," Instrumentation and Measurement, IEEE Transactions on , vol.60, no.2, pp.398,407, Feb. 2011

[4] Xu, Q.R.; Paprotny, I.; Seidel, M.; White, R.M.; Wright, P.K., "Stick-On Piezoelectromagnetic AC Current Monitoring of Circuit Breaker Panels," Sensors Journal, IEEE , vol.13, no.3, pp.1055,1064, March 2013

[5] Suryadevara, N.K.; Mukhopadhyay, S.C.; Kelly, S.D.T.; Gill, S.P.S., "WSN-Based Smart Sensors and Actuator for Power Management in Intelligent Buildings," in Mechatronics, IEEE/ASME Transactions on , vol.20, no.2, pp.564-571, April 2015

[6] Gutierrez, J.; Villa-Medina, J.F.; Nieto-Garibay, A; Porta-Gandara, M.A, "Automated Irrigation System Using a Wireless Sensor Network and GPRS Module," Instrumentation and Measurement, IEEE Transactions on , vol.63, no.1, pp.166,176, Jan. 2014

[7] Ferro, E.; Brea, V.M.; Cabello, D.; Lopez, P.; Iglesias, J.; Castillejo, J., "Wireless sensor mote for snail pest detection," in SENSORS, 2014 IEEE , vol., no., pp.114-117, 2-5 Nov. 2014

[8] ZigBee Alliance, URL: http://www.zigbee.org

[9] IEEE 802.15.4, URL: http://standards.ieee.org/about/get/802/802.15.html

[10] XBee, URL: http://www.digi.com/xbee/

[11] Lay-Ekuakille, A.; Vergallo, P.; Giannoccaro, N.I.; Massaro, A.; Caratelli, D., "Prediction and validation of outcomes from air monitoring sensors and networks of sensors," inSensing Technology (ICST), 2011 Fifth International Conference on , vol., no., pp.73-78, Nov. 28 2011-Dec. 1 2011

[12] Mukhopadhyay, S.C., "Wearable Sensors for Human Activity Monitoring: A Review," in *Sensors Journal, IEEE* , vol.15, no.3, pp.1321-1330, March 2015

[13] Lay-Ekuakille, A.; Vergallo, P., "Decimated Signal Diagonalization Method for Improved Spectral Leak Detection in Pipelines," in Sensors Journal, IEEE , vol.14, no.6, pp.1741-1748, June 2014

[14] Hsiao-Hsien Lin; Hsi-Yuan Tsai; Teng-Chieh Chan; Yen-Shuo Huang; Yuan-Sun Chu; Yu-Chieh Chen; Tai-Shan Liao; Yao-Min Fang; Bing-Jean Lee; Huang-Chen Lee, "An open-source wireless mesh networking module for environmental monitoring," in Instrumentation and Measurement Technology Conference (I2MTC), 2015 IEEE International, pp.1002-1007, 11-14 May 2015

[15] DigiMesh, URL: http://www.digi.com/technology/digimesh/

[16] Kumar, A; Kim, H.; Hancke, G.P., "Environmental Monitoring Systems: A Review," Sensors Journal, IEEE , vol.13, no.4, pp.1329,1339, April 2013

[17] Huang-Chen Lee, "Towards a general wireless sensor network platform for outdoor environment monitoring," Sensors, 2012 IEEE , vol., no., pp.1,5, 28-31 Oct. 2012

[18] MeshBee, URL: http://www.seeedstudio.com/wiki/Mesh_Bee

[19] Jennic JN5168, http://www.tw.nxp.com/products/microcontrollers/wireless_microcontrollers/JN5168.html

[20] Swift01, https://www.kickstarter.com/projects/1279986649/swift01-open-source-mesh-networking-by-swiftlet-te

[21] RF Monolithics XDM2510H, URL: www.rfm.com/products/data/xdm2510hc.pdf

[22] Laird Technologies ZB2430, URL: http://www.lairdtech.com/

[23] CEL FreeStar Pro ZFSM-201-1, URL: www.cel.com/pdf/datasheets/zfsm_201_1_ds.pdf

[24] FabFi, URL: https://code.google.com/p/fabfi/wiki/WikiHome

[25] B.A.T.M.A.N , URL: https://www.open-mesh.org/projects/batman-adv/wiki/Wiki

[26] Sukun Kim, Rodrigo Fonseca, Prabal Dutta, Arsalan Tavakoli, David Culler, Philip Levis, Scott Shenker, and Ion Stoica. 2007. Flush: a reliable bulk transport protocol for multihop wireless networks. In Proceedings of the 5th international conference on Embedded networked sensor systems (SenSys '07). ACM, New York, NY, USA, 351-365.

[27] Jeongyeup Paek and Ramesh Govindan. 2010. RCRT: Rate-controlled reliable transport protocol for wireless sensor networks. ACM Trans. Sen. Netw. 7, 3, Article 20 (October 2010), 45 pages.

[28] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, Maria Kazandjieva, David Moss, and Philip Levis. 2013. CTP: An efficient, robust, and reliable collection tree protocol for wireless sensor networks. ACM Trans. Sen. Netw. 10, 1, Article 16 (December 2013), 49 pages.