# A Transmission Power Optimisation Algorithm for Wireless Sensor Networks

Visham Ramsurrun(✉), Panagiota Katsina, Sumit Anantwar, Amar Seeam, and Sheik Muhammad Arshad Mamode Cassim

School of Science and Technology, Middlesex University, Coastal Road, Flic en Flac, Mauritius
{v.ramsurrun,p.katsina,s.anantwar,a.seeam}@mdx.ac.uk,
sm390@live.mdx.ac.uk

**Abstract.** Wireless sensor networks (WSN) consist of a collection of independent sensor nodes that monitor certain conditions in different locations and transfer data through a network to an endpoint called base station. Because WSN nodes are often deployed outdoors where a direct power outlet may not be available, energy efficiency becomes a key factor in the design of WSNs. This work proposes an antenna transmission power optimisation algorithm (TPOA) which dynamically adjusts the power level of the transmission according to the last received signal strength indicator (RSSI). The mechanism was implemented and tested on an Xbee WSN that uses a modified Mixed Hop and signal Received routing for mobile Wireless Sensor Networks (MHRWSN) protocol. The energy consumption results have shown that the proposed system can save as much as 76% of power in *Tx* operations while maintaining a packet reception rate of above 85% .

**Keywords:** Wireless sensor networks · Transmission Power Optimisation Algorithm · Received Signal Strength Indicator · Xbee · MHRWSN

## 1 Introduction

WSN have applications in various fields including environmental monitoring in agriculture, disaster prevention, patient monitoring in healthcare, and machine monitoring in the industrial sector. WSNs are scalable systems that can consist of a few nodes to several thousands. As such, data routing optimisation is key in large systems to save energy, unlike small systems where communication is commonly achieved through flooding. A WSN may consist of any number of nodes scattered in an area that transfer data to a base station (sink node) through a multi-hop network. A node may consist of any number of sensors, a microcontroller, a battery and a radio transceiver that collect and routes data to a sink node [1]. The sink node is where all sensor data is gathered and it may also provide node management, data processing, and visualisation capabilities to end users. Since sensor networks are often deployed in hard-to-reach (for e.g. under a bridge, on a pole) or downright hostile environments (for e.g. underwater monitoring), a node should have long lifetime while being powered solely by batteries that usually cannot be


R. Zitouni et al. (Eds.): AFRICOMM 2020, LNICST 361, pp. 74–85, 2021.
https://doi.org/10.1007/978-3-030-70572-5_5

replaced or recharged. Consequently, power efficiency has always been one of the major design considerations in sensor networks despite major improvements in computation capabilities in the last decade.

Routing algorithms are an essential element in WSNs as they are responsible for the process of selecting best paths for sending packets from one source to a destination. In adhoc networks like WSNs, nodes have to work together in order to determine the best path for the packets. Different categories of WSN routing algorithms have been developed such as active (for e.g. AODV, TORA and SEER), proactive (for e.g. LEACH, GEAR, OLSR, DSDV, RPL and GPS) and hybrid (which works as active and proactive) routing protocols [2, 3]. The aims of these algorithms are to maximize network lifetime, maintain high packet delivery ratio, minimize delay, manage link failures and optimize reliability and overall Quality of Service (QoS). An even more advanced type of routing algorithms that has emerged of the years is the mobility-based routing protocol that can be used in mobile WSNs and Vehicular Ad-hoc Networks (VANETs). One example of such an algorithm is the Mixed Hop and signal Received routing for mobile Wireless Sensor Networks (MHRWSN) [4]. This paper presents an enhanced and energy-efficient version of the MHRWSN algorithm.

### 1.1 Rationale and Aims

As sensors become cheaper many manufacturers have adopted open source hardware design and standards such as XBee and Zigbee [5]. As a result, transmission power optimisation at the physical layer became viable. Because sensor transceivers share the Industrial, Scientific and Medical (ISM) band with many other devices, signal strength varies widely over time. As a design principle, sensor transceivers are usually set up to transmit at the highest power level to avoid errors and packet loss even when the band is uncongested and irrespective of the distance between nodes. Thus, there is a possibility for power usage optimisation by adjusting power levels based on external network conditions.

The goal of this paper is to implement a dynamically-adjustable power usage mechanism for mobile and static multi-hop networks. The main contributions of our solution are as follows:

- Create a novel RSSI-based transmission power optimisation mechanism that adjusts the transmission power of sensor nodes based on network conditions
- Achieve power savings while maintaining a high packet reception rate and without introducing considerable overhead

The rest of this paper is organized as follows. Section 2 gives a brief review of related works. In Sect. 3, we presents the experimental algorithms and the hardware design that were developed. Section 4 highlights the actual hardware/software implementation setup that was devised. Results and analysis are discussed in Sect. 5. Finally, Sect. 6 provides the conclusion for this paper.

## 2 Related Work

Ebert et al. [6] studied the relationship between antenna power level and energy consumption of IEEE 802.11 interfaces. They found a strong correlation between RF power levels, energy consumption and data rate while the interface was in transmit mode (Tx). A 1 mW to 50 mW increase in power level increased power consumption by 26% while in Tx mode, thus having a major impact on total energy consumption. Power consumption while in receive mode (Rx) remained unaffected.

Carmona et al. [7] made attempts to optimise power level of sensor nodes based on network conditions. They proposed a power management algorithm that optimised transmission power based on link status indicator (LQI). The algorithm was implemented at the application layer, thus not requiring any change in lower layer protocols. The energy efficiency of the system was estimated by using a mathematical model. A 12% and 22% energy savings have been observed on two different radio modules, with the latter having more varied power levels.

Furthermore, Srinivasan and Levis [8] argued that received signal strength indicator (RSSI) correlates well with packet reception rate (PRR) at high transmission power (Tx) when it is above −87 dBm while LQI of 85 could indicate anything between 10% and a 100% packet reception rate. Thus, LQI does not outperform RSSI in all circumstances as it may have been suggested in other works.

Ferro and Velez [4] proposed a mixed hop count and received signal strength routing protocol (MHRWSN) for mobile wireless sensor networks. They assumed that paths to a sink node are frequently being created and destroyed as nodes may be moving closer or farther from one another. The protocol uses a combination of RSSI and hop count to find the best path for data packets. This approach increases packet reception rate and decreases end to end delay.

Pariselvam et al. [9] developed an RSSI-based low energy utilisation scheme to enhance the network lifetime of the Accidental on Demand Distance Vector (AODV) routing protocol. The RSSI level of every node within the route way was checked to find the optimum path to the target in such a way as to minimize the energy expended during the routing process. This new accidental on Demand Distance Vector with the Low Energy Utilization (LE-AODV) is shown to outperform other protocols, namely AOMDV, AOMR and FF-AOMDV, in terms of packet delivery ratio, highest throughput and lowest energy consumption.

## 3 Proposed Design

### 3.1 The Transmission Power Optimisation Algorithm (TPOA)

The algorithm works as follows: when a node R1 transmits a data packet to another node R2 in the network, R2 records the RSSI of the received signal and sends an acknowledgement along with the RSSI value to R1. The RSSI is compared to a threshold and R1 readjusts its power level accordingly. If the node R1 does not receive an acknowledgment after a certain amount of time, it increases its power level to its maximum value and retransmits. This protects R1 from further packet loss due to low power transmission. The system will try to maintain a packet reception rate (PRR) greater than 0.85, the value

where energy loss due to retransmission is much greater than transmission at maximum power level (PL). The TPOA assumes that a path to sink is known. It operates in a multi-hop setting on a link by link basis, while the routing algorithm will be tasked to finding a path to the sink node and appropriately assign the next hop address of the remote nodes. In the algorithm, acknowledgement packets are modified to report RSSI value of the packet that is being acknowledged. Since most remote nodes use duty cycling to save energy, this needs to be taken into consideration. Furthermore, because the sink node is assumed to be also connected to a battery, it also goes into an idle mode to save power. However, it always transmits at maximum power level.

```
Remote Node
upon node R1 goes into an active state
    if node R1 has data to send
        transmit data packet to neighbour
        wait for acknowledge from neighbour
            if acknowledgement is received
                if RSSI value<=upper_threshold and RSSI>=lower_limit
                    wait x seconds and go back into idle state
                else if RSSI value> upper_threshold
                    if PL < maximum PL
                        set PL to PL+1
                        wait x seconds and go back into idle state
                    else
                        wait and go back into idle state
                else if RSSI value< lower_threshold
                    if PL < Minimum PL
                        set PL to PL-1
                        wait  x seconds and go back into idle state
                    else
                        wait x seconds and go back into idle state
            else
                set PL to maximum value
                    retransmit
                    wait x seconds and go back into idle state
    if node R1 receives a data packet from a neighbour
        record sense data and RSSI R
        send an acknowledgement with the recorded RSSI value R to neighbour
        wait x seconds and go back into idle state
end upon
```

**Fig. 1.** Pseudocode of remote node with duty cycling

```
Sink Node
upon sink node goes into an active state
    upon base station receives data packet from a remote node R1
        record sense data and RSSI R
        send an acknowledgement with the recorded RSSI value R to R1
    end upon
end upon
```

**Fig. 2.** Pseudocode of sink node

The sink node is similarly tasked to record the RSSI value and report it back in the acknowledgement packet. The pseudocode of the TPOA for the remote node and sink nodes are shown in the Fig. 1 and Fig. 2 respectively.

### 3.2 The Modified Mixed Hop Count and RSSI Based Routing Algorithm (MHRWSN)

As shown in Fig. 3(a), the idle state of the MHRWSN routing protocol was modified to incorporate the TPOA when transmitting and forwarding data packets. Since it is implemented in the idle state, it will not affect the path selection mechanism of the routing protocol.

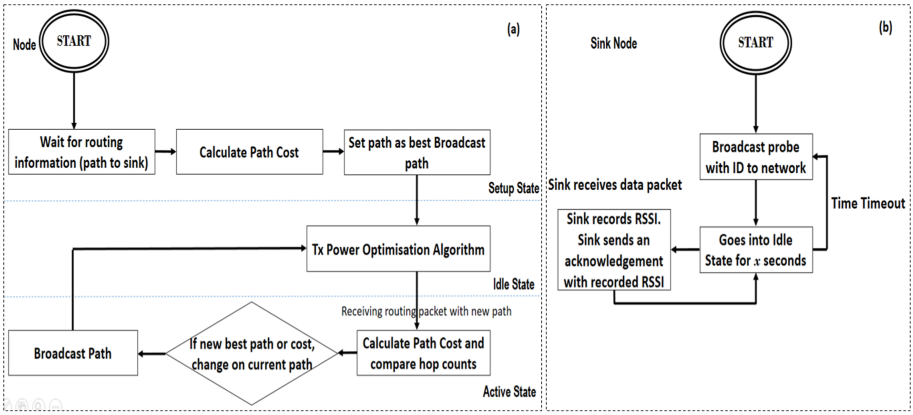


**Fig. 3.** Flowcharts of MHRWSN with TPOA for (a) Sensor node (b) Sink node

All broadcasts of routing packets are still performed at the maximum *Tx* power. The TPOA will only try to optimise the *Tx* power when sensor data is being transmitted. On the other hand, as shown in Fig. 3(b), the sink node will periodically broadcast a packet with its sink ID to the network. This not only permits sensor nodes to initiate their paths, but it causes a periodic refresh of available paths. New paths are identified and installed based on recorded RSSI values, while paths that have ceased to exist are removed.

### 3.3 Hardware Design

The nodes consist of Xbee Series 1 radio modules that were mounted on Arduino Uno microcontrollers. Even though Arduino microcontroller are not known for their energy efficiency, this setup enabled rapid iterations in both the hardware and software design. The Sparkfun Xbee Shield was used to mount the Xbee radio on the Arduino. The Xbee shield greatly simplified the connection between the radio module and the Arduino radio as it mounted directly onto the microcontroller and did not require any additional connection. A testbed was designed to easily debug and assess the system. The LCD was used to display information such as the current power level, the total number of packets sent since system was started, and a message when a packet was lost (acknowledgement timeout) (Fig. 4).

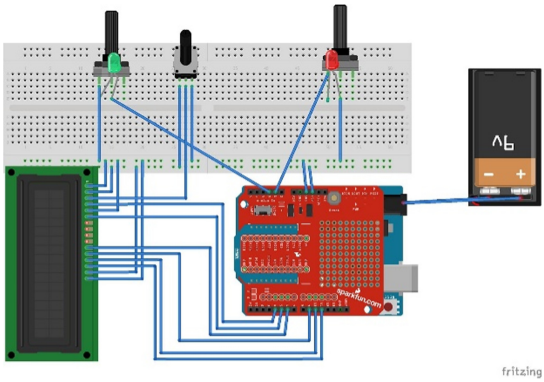


**Fig. 4.** Testbed with an Arduino UNO, LCD, Xbee radio module, Xbee Shield, a 9V battery and breadboard

# 4 Implementation

## 4.1 Software Implementation

The following programming platform, libraries and other software were used during the implementation:

- *Arduino IDE:* The Arduino IDE is an open source programming platform that is used to program Arduino microcontroller in C/C++ languages.
- *Software Serial Library:* This library is used by the Arduino to interface with the Xbee and the LCD. Software serial replicates the functionality of hardware serial.
- *Arduino Xbee Library by Andrewrapp:* The Arduino Xbee library is used by the arduino to communicate with the Xbee. The library supports most Xbee packet types [10].
- *LCD library:* This library allows the Arduino to display information on LCD display with the HD44780 controller [11].
- *XCTU:* XCTU is a software that allows programmers to interact with Xbee RF module. It can interface directly with Xbee radios using AT commands with the AT console GUI. It is also used to configure Xbee radios [12].
- *AT commands:* AT commands are used to send instructions to Xbee Radios like *ATDB*, *ATWR* and *ATPL* [13].

***Node Implementation:*** The TPOA is first implemented in a simple node to node setup. The payload is instantiated as a *unit8_t* that represents the byte type in C. The command *tx.setOption(DISABLE_ACK_OPTION)* disables the acknowledgement of all *Tx* requests that are being made. The acknowledgement packet is modified to include the RSSI value. Thus, we disable the regular acknowledgement of *tx16* packets. The node constantly sends data with the function *xbee.send(tx)* and waits for the modified acknowledgement with the function *rx_packet()*. In the function *rx_packet()*, the node waits for an acknowledgement for 5 s. If it does not receive a packet during that time frame, it assumes the packet was lost. On the other hand, if it receives a packet, it records the RSSI in the

packet payload. The algorithm also uses the following two values: *Upper_threshold* is equal to −87 dBm whereas *Lower_threshold* is equal to −80 dbm (A lower RSSI represents a better signal). The algorithm checks the RSSI and compares it to a lower and upper threshold. It then sets the power level accordingly with the *sendAtCommand()* function.

***Sink Implementation:*** The sink does not optimise its transmission power as it may communicate with many remote nodes. It only sends a modified acknowledgment with the RSSI value of any received packet. In this configuration, the sink waits for a packet from a remote sensor node. When a packet is received, it reads the RSSI value from the Xbee radio with the function *getRSSI()*. It then sends a packet with the RSSI as a payload. The sink will not send a regular acknowledgement as the received packet will have the acknowledgement *option(byte)* disabled. The *getRSSI()* function executes the *ATDB* command.

***MHRWSN Implementation:*** The routing algorithm is also implemented using the Arduino IDE according to the specifications given by Ferro and Velez [4].

***Xbee Radio Configurations:*** The Xbee radio settings like *Pan ID*, *16-bit source addresses* and *API Enable,* are applied using XCTU.

### 4.2 Hardware Implementation

Five Xbee modules mounted on Arduino Uno microcontrollers with Sparkfun Xbee shields were made.

***Xbee S1 Module with* 1 mW *Wire Antenna:*** The Xbee Series 1 is one of the most popular 802.15.4 transceivers. Although it is marketed as an 802.15.4 device, it only uses the lower layer of Zigbee and does not support Zigbee routing protocol out of the box. However, the Xbee module can use a serial port to interface with other devices. By using an Arduino to perform network layer operation, we can implement our own routing algorithm [14].

***The Sparkfun Xbee Shield:*** It greatly simplified the communication between the Xbee module and the Arduino. The shield also provides a prototyping area and a serial switch to select between hardware (UART) serial and software (DLine, Arduino Rx and Tx pins) serial. The shield, however, requires headers to be stacked on an Arduino. We therefore soldered the Arduino R3 headers onto the shield with rosin core solder. The headers include a 6-pin, 2 × 8-pin, and 10-pin header [15].

***The Arduino Uno Microcontroller:*** The microcontroller was tasked to run both the transmission optimisation algorithm and the MHRWSN routing algorithm. It communicated with the Xbee with its Tx and Rx pin. The Xbee S1 is just a transceiver, and hence, cannot be programmed to run the algorithm. The microcontroller only interfaced with the Xbee transceiver during these three operations: packet reception, packet reception and AT command.

***LCD Screen:*** The LCD used was a HD44780 standard 2 × 16 lines display with white characters on a blue background. It operates at 5v and both its backlight and contrast can be adjusted. It also support 4 and 8 bit parallel interface.

**10 KΩ *Pot, Breadboard and Battery:*** The 10 KΩ pots were used to step down the 5V voltage from the digital pins and 5 V pin of the Arduino. They were also used to control the contrast of the LCD screen. A breadboard was used to construct the circuit, together with a 9 V battery with a battery holder for supplying power to the Arduino.

## 5 Testing, Results and Discussion

To evaluate our TPOA, tests were performed for a multi-hop setup using MHRWSN routing algorithm, with sensor nodes being at two different distance intervals from the sink node, namely 30 m and 70 m. A preliminary pilot test was carried out and it showed distinctly observable and recordable changes in RSSI readings when at 30 m and 70 m respectively. The results were then compared with a setup using fixed power levels. The sink node remained at a fixed location while the location of the other nodes changed after every $n = 50$ packets. The test was performed in an outdoor environment. The packet reception rate (PRR) was recorded after transmitting $n$ packets. A packet is considered lost if the node does not receive an acknowledgment within a limited time frame. The PRR was given by the equation:

$$\text{PRR} = 1 - (\textit{Total Packet Loss}/\textit{Total Packet Sent}) \tag{1}$$

The RSSI value was recorded directly at the sink node. Every change in power level caused by the optimisation algorithm was recorded. The sink node always transmits at the maximum power level. The TPOA was implemented on the other non-sink nodes.

**Table 1.** Results with nodes at 30 m and 70 m from one another with TPOA

| Distance | 30 m | 70 m |
|---|---|---|
| PRR | 1 | 0.98 (1 packet loss) |
| Average RSSI | −82.36 | −83.408 |
| Total packets transmitted while PL = 0 | 16 | 5 |
| Total packets transmitted while PL = 1 | 29 | 16 |
| Total packets transmitted while PL = 2 | 3 | 26 |
| Total packets transmitted while PL = 3 | 1 | 2 |
| Total packets transmitted while PL = 4 | 1 | 2 |

As we can see in Table 1, using the TPOA, the setup dynamically adjusted the power levels in order to maintain high packet reception rate. At 30 m, the majority of the packets were delivered using Power Levels 0 and 1, whereas at 70 m, the bulk of the packets

**Table 2.** Results with nodes at 30 m and 70 m from one another with fixed PL

| Distance (30 m) | | | | | |
|---|---|---|---|---|---|
| Fixed PL | PL = 0 | PL = 1 | PL = 2 | PL = 3 | PL = 4 |
| PRR | 0.72 (14 packets lost) | 0.88 (6 packets lost) | 1 | 1 | 1 |
| Average RSSI | −89.704 | −86.653 | −76.793 | −69.578 | −60.867 |
| Distance (70 m) | | | | | |
| Fixed PL | PL = 0 | PL = 1 | PL = 2 | PL = 3 | PL = 4 |
| PRR | 0.7 (15 packets lost) | 0.78 (11 packets lost) | 1 | 1 | 1 |
| Average RSSI | −94.790 | −90.802 | −82.380 | −81.591 | −79.592 |

were delivered using Power Levels 1 and 2. From Table 2, we can see that because of the fixed power levels, the setup could not move between power levels to maintain high packet reception rates when needed, thus causing packets to be dropped. As such, we can see the effectiveness of the TPOA in allowing the setup to switch power gears when needed in response to dynamic network conditions so as to maintain high PRR, while using minimum energy.

### 5.1 Power Consumption

Since only the power level was provided by the manufacturer, the power in milliwatts is calculated with the formula [16]:

$$\boldsymbol{P(\mathrm{mW}) = 1\,\mathrm{mW} \cdot 10^{(P(\mathrm{dBm})/10)}} \tag{2}$$

The manufacturer also mentioned that only the maximum power level is calibrated while the others are only approximations (Table 3).

**Table 3.** Power level to mW conversions [16]

| Power level | Power level (dBm) | Power (milliwatt) |
|---|---|---|
| 0 | −10 | 0.1 |
| 1 | −6 | 0.2512 |
| 2 | −4 | 0.3981 |
| 3 | −2 | 0.6309 |
| 4 | 0 | 1 |

The total power consumed by TPOA was compared to power consumed with fixed transmission power for $n = 50$ packets. A graph of Total power consumption by $Tx$ against total packet number was plotted.

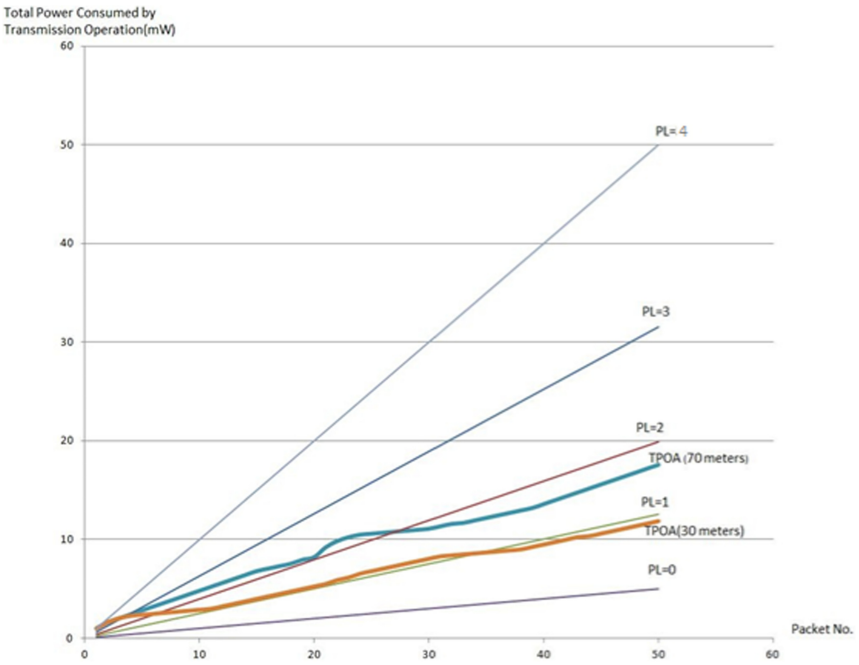


**Fig. 5.** Total power consumption of *Tx* operations

### 5.2 Observations

As shown in Fig. 5, TPOA has a total transmitting power consumption of 17.578 mW at 70 m and 11.8612 mW at 30 m. At 70 m TPOA is comparable to a fixed transmission power of PL = 2 (20.035 mW consumed). It is about 12% more efficient and it does so by maintaining a 0.98 packet reception rate. At 30 m TPOA is comparable to PL = 1 (12.203 mW consumed), but TPOA is only about 3% more efficient. However, the TPOA algorithm has a PRR of 1 while when with fixed transmission power of PL = 1, the PRR is only 0.88. When compared to a system which only transmits at the highest power level available, i.e. PL = 4 using a total of 50.012 mW of power, the TPOA is about 76% and 65% more efficient at 30 and 70 m respectively. Some variations in power consumption were noticed during transmission of packets 20–25. This may be explained by the fact that TPOA may have had to jump to a higher power level to transmit some of the packets because of destructive interference caused by some stray signals in the testing environment. Thus, we can see that TPOA reduced power consumption while keeping a high packet reception rate.

## 6 Conclusion

This paper proposed an effective mechanism to optimise the transmission power of remote nodes according to actual network conditions using RSSI. The system used two threshold values to maintain a high packet reception rate of above 85% so as to avoid

loss of energy due to retransmissions. The system was implemented using Xbee module and Arduino microcontroller. The test results showed that the proposed algorithm can save as much a 76% of power in Tx operations. Future works will involve applying the TPOA in other algorithms like AODV.

**Acknowledgments.** This work was supported by the Mauritius Research and Innovation Council (MRIC) under the URIGS research grant scheme ref. MRC/RUN/1605.

## References

1. Gopika, D., Panjanathan, R.: A comprehensive study on various energy conservation mechanisms in wireless sensor networks. In: 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Vellore, India, pp. 1–5 (2020)
2. Arat, F., Demirci, S.: Energy and QoS aware analysis and classification of routing protocols for IoT and WSN. In: 7th International Conference on Electrical and Electronics Engineering (ICEEE), Antalya, Turkey, pp. 221–225 (2020)
3. Mahakalkar, N., Pethe, R.: Review of routing protocol in a wireless sensor network for an IOT application. In: 3rd International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, pp. 21–25 (2018)
4. Ferro, J.M., Velez, F.J.: Combined hop count and received signal strength routing protocol for mobility-enabled WSNs. In: IEEE Vehicular Technology Conference (VTC Fall), Quebec City, QC, pp. 1–6 (2012)
5. Orgon, M., Zagajec, L., Schmidt, I.: XBee technology: complex evaluation of radio parameters. In: 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Dublin, Ireland, pp. 1–6 (2019)
6. Ebert, J.P., Aier, S., Kofahl, G., Becker, A., Burns, B., Wolisz, A.: Measurement and simulation of the energy consumption of a WLAN interface. TKN technical report TKN-02–010, vol. 314. Technical University Berlin, Berlin, Germany (2002)
7. Carmona, C., Alorda, B., Ribot, M.A.: Energy consumption savings in ZigBee-based WSN adjusting power transmission at application layer. In: 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS) Palma de Mallorca, pp. 1–6 (2014)
8. Srinivasan, K., Levis, P.: RSSI is under-appreciated. In: Third Workshop on Embedded Networked Sensors (EmNets), Cambridge, MA, USA (2006)
9. Pariselvam, S., Manikandan, M., Praveenkumar, P., Vinothvarma, R.: Energy efficient RSSI based low energy node utilization routing In: MANET. IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), Pondicherry, India, pp. 1–6 (2019)
10. Rapp, A., Kooijman, M.: Arduino library for communicating with XBee radios in API mode. https://github.com/andrewrapp/xbee-arduino. Accessed 02 July 2020
11. Perry, B.: hd44780 - Extensible hd44780 LCD library. https://www.arduinolibraries.info/libraries/hd44780. Accessed 02 July 2020
12. Digi International Inc.: XCTU - Next Generation Configuration Platform for XBee/RF Solutions. https://www.digi.com/products/embedded-systems/digi-xbee/digi-xbee-tools/xctu. Accessed 02 July 2020
13. Digi International Inc.: PL (TX Power Level). https://www.digi.com/resources/documentation/Digidocs/90001506/reference/r_cmd_pl.htm?TocPath=AT%20commands%7CMAC%2FPHY%20commands%7C_____5. Accessed 02 July 2020

14. SparkFun Electronics: XBee 1mW Wire Antenna - Series 1 (802.15.4). https://www.sparkfun.com/products/retired/8665. Accessed 02 July 2020
15. JIMB0: XBee Shield HookupGuide. https://learn.sparkfun.com/tutorials/xbee-shield-hookup-guide. Accessed 02 July 2020
16. RapidTables.com: dBm to mW Conversion. https://www.rapidtables.com/convert/power/dBm_to_mW.html. Accessed 02 July 2020