mTOSSIM: A simulator that estimates battery lifetime in wireless sensor networks

J.M. Mora-Merchan [a], D.F. Larios [a], J. Barbancho [a], F.J. Molina [a], J.L. Sevillano [b], C. León [a]

[a] *Department of Electronic Technology, Universidad de Sevilla, Seville, Spain*
[b] *Department of Computer Technology and Architecture, University of Seville, Seville, Spain*

Keywords:

WSN
Simulation
Battery lifetime
Energy consumption

A B S T R A C T

Knowledge of the battery lifetime of the wireless sensor network is important for many situations, such as in evaluation of the location of nodes or the estimation of the connectivity, along time, between devices. However, experimental evaluation is a very time-consuming task. It depends on many factors, such as the use of the radio transceiver or the distance between nodes. Simulations reduce considerably this time. They allow the evaluation of the network behavior before its deployment. This article presents a simulation tool which helps developers to obtain information about battery state. This simulator extends the well-known TOSSIM simulator. Therefore it is possible to evaluate TinyOS applications using an accurate model of the battery consumption and its relation to the radio power transmission. Although an specific indoor scenario is used in testing of simulation, the simulator is not limited to this environment. It is possible to work in outdoor scenarios too. Experimental results validate the proposed model.

1. Introduction

Wireless Sensor Networks (WSNs) have been widely used in many areas [1], such as environmental monitoring and control, localization [2], healthcare and medical research, and national defense and military applications. The main components of a WSN node are as follows:

- A microcontroller that processes data and controls the other components in the node.
- A wireless radio transceiver that combines the radio transmitter and receiver. Typically, these devices use ISM bands, which are internationally reserved for un-licensed use in radio frequency applications (RF) (such as the 173, 433, 868, 915 MHz bands and 2.4 GHz bands). Standard protocols that are specially developed for these kinds of devices have been proposed, such as IEEE 802.15.4 [3], Zig-Bee [4], or ISO/IEC 14543-3-10 [5].
- A power source that provides energy to the electronic device. WSNs usually use batteries or super-capacitors as a power source. Batteries can be either rechargeable or non-rechargeable depending on the application, but in both cases, the energy available is often limited.

A typical node of a WSN is a device with several strong restrictions [6]: low power consumption, low weight, and low cost. Especially in the case of non-anchor nodes or tags, it is important to consider the battery lifetime of these devices, as they have high power energy restrictions. Typically, anchor nodes have rechargeable battery systems, but in order to reduce the

* Corresponding author. Tel.: +34 954556142.
E-mail addresses: jmmora@us.es (J.M. Mora-Merchan), dflarios@dte.us.es (D.F. Larios), jbarbancho@us.es (J. Barbancho), fjmolina@us.es (F.J. Molina), jlsevillano@us.es (J.L. Sevillano), cleon@us.es (C. León).

weight and size, the power supplies of tags are often small and not rechargeable. Typical tags are mobile devices that an animal or a person should carry [7,8].

In WSNs, it is necessary to use low power components to increase node autonomy. So, it is necessary to maintain the microcontroller and the radio transceiver in the idle state as long as possible. This is because radio transceivers are the devices with the highest power consumption in WSNs nodes. Moreover, radio behavior depends on the battery level: Its power transmission decays with the discharge of the batteries (see Section 4).

The radio transmission power can be estimated with the RSSI (Received Signal Strength Indication) parameter. The estimation of this parameter is necessary in many applications, such as determining the best allocation of devices [9], optimizing the network topology [10], estimating the localization of mobile tags [11,12], or evaluating the connectivity between devices [13].

Although power consumption and RSSI can be easily measured, monitoring the behavior of a real network, considering its real lifetime and the attenuation of the transmitted power along the battery discharging, requires a high amount of time. In these situations, simulators would serve as a helpful tool.

In this article, we propose a simulator, called *mTOSSIM* [14], that permits an estimation of the power consumption in a long-term analysis as well as an estimation of the batteries lifetime of the devices, taking the radio power consumption into account. Moreover, the simulator allows an estimation of the RSSI attenuation, considering the effects of battery discharging in the emitter. In addition, indoor scenarios with obstacles between devices are also considered. As an example, in this article, the simulator has been applied for a case study of devices that are powered with supercapacitors.

The rest of the article is organized as follows: In Section 2, a brief introduction about the motivation and the state of the art of simulators applied to WSNs is presented. Section 3 describes the simulation techniques that are used to estimate power consumption. The relationship between RSSI and battery level is discussed in Section 4. Section 5 describes the model that is used for estimating the RSSI attenuation. All this information is used in Section 6 for describing the complete simulation algorithm, which is implemented in the simulator described in Section 7.

This article not only depicts the theoretical results. As described in Section 8, some experiments have been performed to evaluate the simulator. Section 9 presents our conclusions.

2. Motivation and related works

The use of simulators in WSN is very extended, and several interesting open-source simulators have been proposed [15]. However, these simulators mainly focus on the study of the network topology [16]. Another approach that frequently appears in the literature is the use of adhoc simulators for testing specific application algorithms [17]. None of these approaches permit the simulation of the real code deployed over a WSN, nor do they allow the simulation of real applications but only parts of the proposed algorithms.

In order to solve these issues in TinyOS applications (one of the more popular platforms for WSNs), TOSSIM [18,19] (TinyOS Simulator) has been proposed. First, it is a set of components that serves two main purposes: It provides low level components for the TinyOS stack that simulate the behavior of elements which are related to the hardware of physical motes; and second, TOSSIM substitutes low level elements of the operating system using an event-driven engine. The result of using TOSSIM components is a source code in a kind of object-oriented approach in the C Language that simulates WSN applications.

Of course, there are some trade-offs in TOSSIM design [20,19]. The main limitations are that it does not consider the environment of the nodes (e.g. whether the network is deployed indoors or outdoors); and that it does not offer a model which simulates energy consumption and use. Moreover, the radio propagation model used by default in TOSSIM is pretty simple, and there is no assumption about battery drain and its effects on the network.

Several attempts have been made to overcome these failures in the TOSSIM simulator, such as PowerTossim [21] or TOSSIM-T2 [22]. These simulators try to evaluate energy consumption in the nodes. They work in two stages: making a log of every time a component switches on or evaluating the energy involved in each time period through a static analysis at the end of the simulation. This approach suffers from a lack of feedback in the simulation: The available energy does not affect the simulated behavior; whereas in real life, the transmission power of transceivers depends on battery status. Other simulators developed for TinyOS applications (such as ATEMU [23]) suffer from a similar lack of consideration about energy consumption.

One interesting contribution for TOSSIM is SWARM-eTOSSIM [24]. It permits the simulation of the behavior and energy consumption of real applications as well as the modeling of batteries capacity over time. However, it can only test applications based on a swarm library [25], which is a multi-agent framework.

In order to overcome these problems, in this article, we propose mTOSSIM, a simulator that permits the evaluation of real TinyOS applications. It models the power consumption dynamically during simulations, as well as its effect on the behavior of the network. For example, mTOSSIM permits the consideration of the real situation when a node completely discharges its battery, thus changing the network topology, or the evaluation of the nodes that appear disconnected to the network, due to its reduction of transmission power which is caused by the reduction of the power energy available in the battery.

3. Evaluation of the power consumption in a WSN

The power consumption in a WSN can be estimated as a function of its components. Table 1 sums up the power consumption in the telosB node [26].

In general, the energy consumed by the components of a WSN node (such as external sensors, the microcontroler or the radio transceiver) can be modelled as describes the following equation:

$$E_{comp} = P_{comp} \cdot T_{on,comp} = V_{Batt} \cdot I_{comp} \cdot T_{on,comp} \tag{1}$$

where E_{comp} is the energy consumed by the component of the node, P_{comp} is its power consumption, $T_{on,comp}$ is the time while this component remains active and I_{comp} is the current consumption of the component which can be obtained from the components datasheet. For simplicity, the average value of the current is used.

Using this information, the total energy consumption of the node can be estimated by adding the consumption of all the devices, as described by

$$E_{Node} = \sum E_{Comp} \tag{2}$$

The proposed simulator can consider the consumption of an arbitrary number of components, in function of the application. Actually, common WSN applications use low power consumption components and tasks that do not require a heavy local processing. In these cases, the power consumption of the node can be estimated considering only the power consumption of the radio transceiver. This is due to the fact that (as can be observed in Table 1) the power consumption of the radio transceiver is an order of magnitude higher than the other ones. Moreover, the power consumption in the reception mode is similar to the power consumption in the transmission mode. Furthermore, the power consumption in the idle state can be neglected (1000 times lower than the active state).

TelosB node uses a CC2420 radio transceiver (operating in the 2.4 GHz band), from Texas Instruments. According to its data sheet [27], the radio transceiver consumes 18.8 mA (56.4 mW at 3 V) in the reception mode (I_{RX}) and 17.4 mA (52.2 mW at 3 V) in the transmission mode (I_{TX}). As can be observed, Texas Instruments offers more conservative values than the measurements described in [26], considering a 3 V power supply (two 1.5 V batteries). It could be assumed that the real consumption would be between these two estimations. However, in this article, we are going to use the most restrictive estimation, that is, the values ordered by Texas Instruments. Considering these power consumption values, the energy needed to send a message (E_{TX}) can be estimated as $E_{TX} = T_{TX} \cdot P_{T_X}$; whereas the energy needed to receive a message (E_{RX}) can be obtained with $E_{RX} = T_{RX} \cdot P_{TX}$. Where T_{TX} is the time used to send the message, and T_{RX} is the time while the radio is in the reception mode.

P_{TX} and P_{RX} can be obtained as a function of the current consumption of the radio transceiver. In telosB, it can be obtained as described in

$$\begin{aligned} P_{TX} &= V_{Batt} \cdot I_{TX} = V_{Batt} \cdot 17.4\ \text{mA} \\ P_{RX} &= V_{Batt} \cdot I_{RX} = V_{Batt} \cdot 18.8\ \text{mA} \end{aligned} \tag{3}$$

where the battery voltage V_{Batt} varies as a function of the technology used and its level of charge (β). For lithium batteries, Peukerts equation can be used considering the restrictions on discharge current and temperature imposed by this model [28].

Using supercapacitors, a model based on the ideal behavior of a capacitor can be used [29], such as described in

$$V_{Batt} = \sqrt{\frac{2 \cdot E}{C}} = \sqrt{\frac{2 \cdot E_{Batt} \cdot \beta}{C}} \tag{4}$$

where E_{Batt} is the maximum energy stored in the capacitor (nominal value), expressed in Joules; C is its capacity; and β (in [0, 1] range) represents the energy remaining in the battery per unit.

4. Evaluation of the relationship between battery level and power transmission

As commented on earlier, there is a relationship between power transmission $P_{TX(\text{dB})}$ and energy battery level. In devices with non-rechargeable batteries (non-anchor nodes), as described in some papers [30,31], power transmission decreases as battery level decreases.

It is common to assume a lineal decreasing model in the operating range of the battery [32]. Accordingly, considering a not completely discharged device, the power attenuation would be modeled as follows:

Table 1
Typical telosB power consumption [26].

Parameter	Value
Microcontroller active power (P_{micro})	3 mW
Radio transceiver receive power (P_{RX})	38 mW
Radio transceiver transmit power at 0 dBm (P_{TX})	35 mW
Radio transceiver power in idle mode	3 µW
Temp sensor (worst case)	2.75 mW
Total active power	41 mW

$$P_{TX(\mathrm{dB})} = P_{TX(\mathrm{dB}max)} - P_{Att(\mathrm{dB})} \cdot (1 - \beta) \tag{5}$$

where $P_{TX(\mathrm{dB}max)}$ is the maximum power transmission in dB (0 dBm) in telosB nodes, and $P_{Att(\mathrm{dB})}$ is the power loss coefficient, which is expressed in dB.

As described by other authors [31], with lithium batteries, $P_{TX(\mathrm{dB}max)}$ would decrease up to 10 dB in the voltage operative range of the device. Using supercapacitors, as described in Section 8.1, RSSI decays up to 3 dB.

This proposed lineal decreasing model is only applicable while the battery level is working over a minimum threshold (which varies depending on the hardware platform). If the battery level is lower than this threshold, no message is sent and $P_{TX(\mathrm{dB})} = -\infty$ dB.

5. Path loss model for indoor applications

In the literature, many propagation models have been proposed, such as Friis Path Loss Model [33] or ground reflection model [34]. In general, these models do not offer a good estimation for indoor applications. Reflections, walls, and floor attenuation drastically modify the behavior of transmission signals in indoor applications, especially at high frequencies such as 2.4 GHz that is used in 802.15.4 devices.

For these situations, ITU [35] offers a model that takes these issues into account. The model is commonly called “ITU model for indoor attenuation” and expresses the attenuation path loss in dB (L), such as:

$$L = 20\log(f) + N\log(d) + P_f(n) - 28 \tag{6}$$

where f is the frequency of transmission in MHz; d is the distance in meters; N is the distance power loss coefficient; n is the number of floors or walls between the transmitter and receiver; and $P_f(n)$ is the floor/walls loss penetration factor. Attending to [35], the values for N and $P_f(n)$ in 2.4 GHz are summarized in Table 2.

Using this model, it is possible to estimate the attenuation of the system. Some experiments have been reported in the literature that evaluate this model, especially for the Wi-Fi connection [36] in the 2.4 GHz band; the same frequency is used in the 802.15.4 transceiver. These results as well as our own experiments, described in Section 8, validate this model as a good approximation for indoor uses.

Other alternative models provide a more realistic estimation of the path loss for indoor applications, but they require complex ray models [37]. Moreover these models require indepth knowledge about the environment, which is usually difficult to obtain.

In this article, the ITU model is used, as it offers a good trade-off between complexity and model accuracy.

Reflection effects are very difficult to model, especially with regard to indoor uses. For this reason, reflection is modeled as a noise added to the signal. In this case, a Gaussian noise is considered. Standard deviation can be obtained empirically. Attending to our tests, a standard deviation of 5 dB is a good approximation for applications, where the sensor networks share the channel with Wi-Fi routers. Without sharing the channel, a lower value can be used.

According to this, the received signal level ($RSSI_{\mathrm{dB}}$) can be estimated as follows:

$$RSSI_{\mathrm{dB}} = P_{RX(\mathrm{dB})} + P_{noise} - L \tag{7}$$

where P_{noise} is the noise added for modeling the reflection, which is expressed in dB.

6. Proposed simulation technique

The simulator developed uses the models described in the previous sections for estimating WSNs in indoor applications. For obtaining the transmitted power attenuation, it is necessary to know the battery level (β). In order to do this, it is necessary to obtain the power energy consumption in both the transmission (E_{TX}) and reception (E_{RX}) modes.

The power energy consumption in the transmission mode varies as a function of the length (number of bytes) of the message, which depends on the overhead of the system. 802.15.4 defines four classes of messages (Fig. 1). The overheads of these messages are (in bytes):

- Data frame: $N_{total} = 11 + l_{addr} + n$, where n is the length of the payload, and l_{addr} is in [4–20] range. n varies depending on the application.
- ACK frame: $N_{total} = 11$.
- Beacon frame: $N_{total} = 11 + l_{addr} + n$, with n in the [4–10] range, and n varies depending on the application.

Table 2
Typical values of N and $P_f(n)$ in 2.4 GHz.

Parameter	Residential area	Office area	Commercial area
N	28	30	22
$P_f(n)$	$4 \cdot n$	$15 + 4 \cdot (n - 1)$	$6 + 3 \cdot (n - 1)$

- MAC command frame: $N_{total} = 12 + l_{addr} + n$.

Independently of the type of frame, the total length should be less than or equal to 133 bytes (127 bytes of the MAC and application layer plus 4 bytes of the sync header and 1 byte of the PHY header).

Using this information and according to Section 3, the energy consumption when sending a frame (E_{TX}) could be obtained as described in

$$E_{TX} = N_{total} \cdot T_{byte} \cdot P_{TX} \tag{8}$$

where T_{byte} is the time required to send a byte. In the reception mode, the energy consumption can be obtained as described in

$$E_{RX} = T_{RX} \cdot P_{RX} \tag{9}$$

P_{RX} can be obtained as a function of the current battery voltage, as described in Section 3. β should be updated after every event, with an event being a transmission or a reception event.

After a reception, the battery level should be updated according to the following equation:

$$\beta' = \frac{\beta \cdot E_{Batt} - E_{RX}}{E_{Batt}} = \beta - \frac{E_{RX}}{E_{Batt}} \tag{10}$$

where E_{Batt} is, again, the nominal value of the battery in Joules.

In transmission events, the system first tests whether it has enough battery to send a message (β is higher than a certain threshold, which varies depending on the platform). If this is the case, the device updates its power transmission power (P_{TX}), according to the equation described in Section 4. After the message is sent, β is updated, according to Eq. (11).

$$\beta' = \beta - \frac{E_{TX}}{E_{Batt}} \tag{11}$$

A node only receives a message if its power energy (RSSI) is higher than a minimum threshold. TelosB nodes have a typical sensitivity of -95 dBm.

7. Description of mTOSSIM

The proposed tool, called *mTOSSIM*, is a set of applications that automates the creation of ad hoc simulators for TinyOS networks which operate over TOSSIM.

TOSSIM translates source code of TinyOs application (in NesC language) to a language C file called "*app.c*". "*app.c*" file contains all code related with each mote behaviour (including information on component state). Network and therefore the simulation behaviour depends on file "*driver.c*" which defines network topology, control simulation, shows diagnostics info and results, and the interface with users. The name conventions used follows [38] ones. Although many patterns help to simplify design and development of driver.c file, its contents are quite specific on the working application. mTOSSIM provides a "*driver.c*" file that can be easily customized.

mTOSSIM is also a GUI that integrates its own tools with TinyOS chain tool and is an interactive viewer of simulation logs as well as an exporter of logs. As a chain tool, Fig. 2 shows the relationship between different components in play. There are three main functionality areas in the architecture: Network Topology, Simulation Behavior, and Simulated Application.

Network Topology in the simulation is described by attenuation between each couple of nodes (Eq. (6)). Provided *"driver.c"* reads information on network topology from *"net.def"* file. In *"net.def"* file, the core of this area, there is also information about simulations, such as switch on time or localization of each node. To facilitate the processing of that data, *"net.def"* is, in fact, a x-macro definition file [39]. There are two ways of creating that file: parse of a more human friendly file (*"net.map"*) with Perl script *"net.pl"* or generating it from mTOSSIM GUI. The former is a more powerful method, but the file has to be written manually; the latter is easier to use. Our way of work involves creating a template from GUI and later, adapting it by hand (if necessary) to facilitate further automation in the simulation.

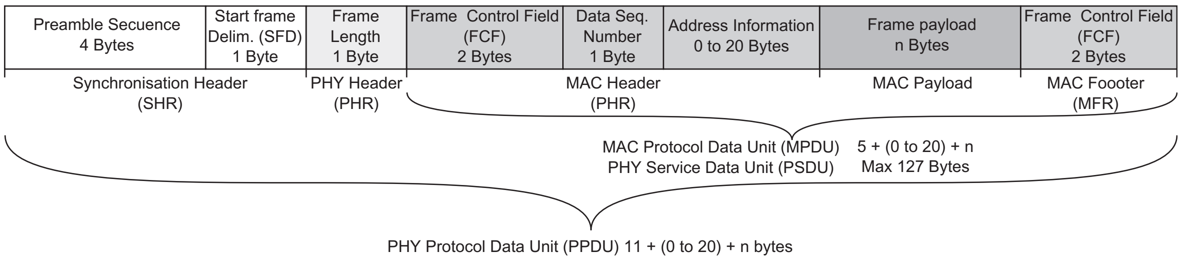


Fig. 1. 802.15.4 PPDU.

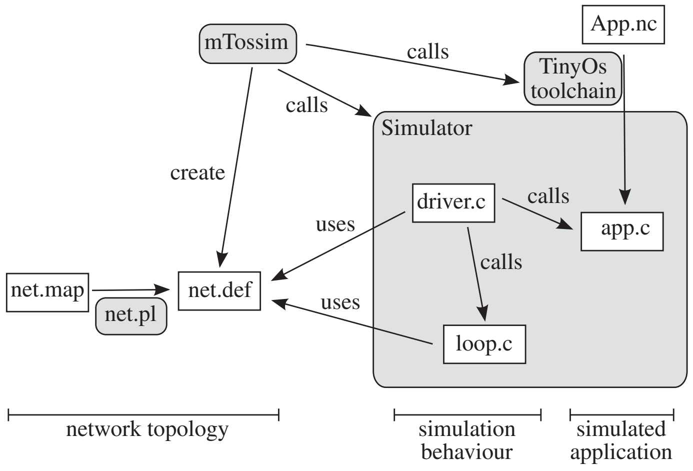


Fig. 2. mTOSSIM architecture.

The core of the simulator code area manages the properties of the simulation, the behavior of the system, and the information that is collected and shown. It is here where our model of battery and its effect on the quality of inter-nodes communication is integrated. The log system of TOSSIM is fully integrated with mTOSSIM. It permits monitoring not only specific information related to simulated applications such as local variables but also all TinyOS or TOSSIM internal services, such as private system variables or state of different subsystems (transceiver, sensors or micro-controller).

Finally, the Simulated Application area is where the real TinyOS application is simulated. The architecture of mTOSSIM has as its main objective the simulation of the code without altering the WSN application or the TinyOS source code; so, this code can be really used with real nodes without modification and test results in real life. Although the mTOSSIM tool chain does not fully comply with this requirement yet, the code in "*app.nc*" is quite isolated from the rest with the help provided by the C preprocessor.

mTOSSIM permits the creation of network topologies, to compile simulators and to examine the results of simulation. Currently, it does not run as an interactive simulator but as an interactive log viewer: The simulation behavior is determined at compile time. Then, mTOSSIM GUI allows the navigation around logs (Fig. 3), moving through time in the simulation.

7.1. Battery simulation

mTOSSIM permits to dynamically estimate platforms by estimating the effects of the battery discharge on the network.

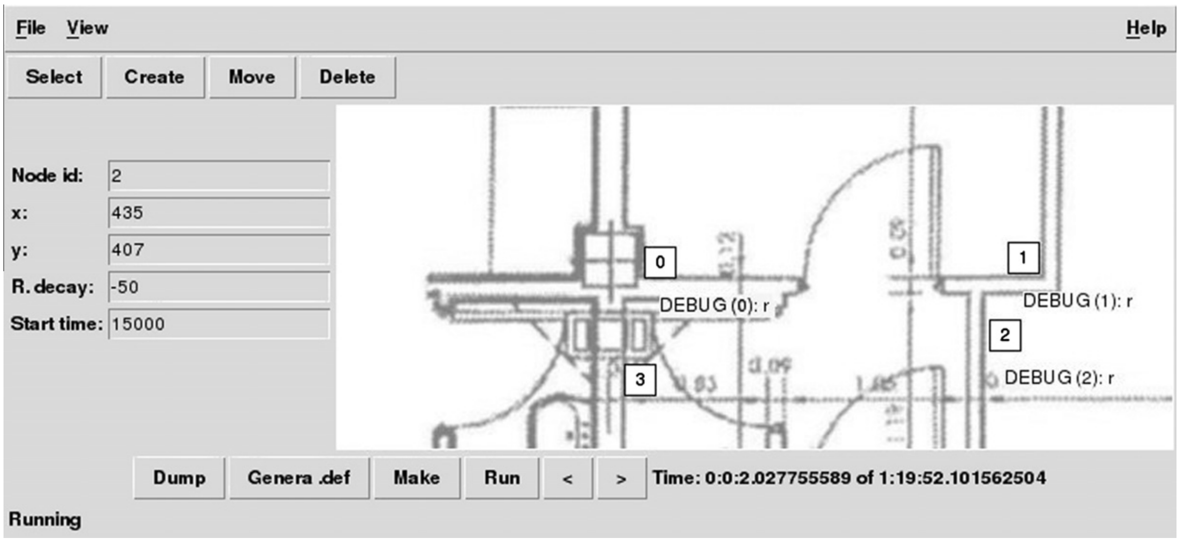
TOSSIM is an event-driven simulator. This means that the simulator determines which events occur and at what time instant. The possible effects of an event are a change of state of simulator or the creation of a new event that is executed in the future. The state of the simulator does not change between two consecutive events. TOSSIM offers the `runNextEvent()` function with the aim of waiting for the next scheduled event and for executing the planned computation associated with the event.

mTOSSIM acts over the TOSSIM event loop. Between every event, mTOSSIM reads simulation status, updates battery level, and modifies simulation status accordingly. Some assumptions have been made: The only peripheral used is the transceiver. The transceiver power consumption in both the RX mode and the TX mode is the same (this is almost true in telosB motes). Transmission power is always 0 dBm. Finally, microprocessor power consumption is negligible, as it is several orders of magnitude lower than the transceiver and, in our tests, lower than capacitor auto-discharge rate.

The battery model follows Eq. (4). We need to know the transceiver status to update the battery level. A full main structure of the update battery is shown in Listing 1. The code follows the same convention as the TOSSIM generated code: Use an array for each status variation of the node.

The structure is the following: lines 2–5 acquire information on the TOSSIM simulator (current node active, transceiver status, and current time). Lines 7–18 update the battery status. `beta` update is done following Eq. (10). `batteryJ2V(float beta)` is where the battery (in our case, a capacitor) is modeled. This function converts beta values to volts (our model is shown in Eq. (4), see also Listing 2).

Finally, lines 21–26 save TOSSIM and mTOSSIM statuses. TOSSIM works as a static network simulator. The network conditions are stable until something alters them. To simulate an evolution in the network (either battery drop or movements of nodes), it is necessary to rewrite all connections between the nodes. Function `newpow` (shown in Listing 3) updates network connections.



Node id: 2
x: 435
y: 407
R. decay: -50
Start time: 15000

Time: 0:0:2.027755589 of 1:19:52.101562504

Fig. 3. mTOSSIM snapshot.

7.2. RSSI vs. battery

There are some limitations in the TOSSIM model while working with RSSI values. In "real" code (code to program telosB motes), the RSSI value is obtained from a message received with

```
RSSI=(uint16_t) call
    CC2420Packet.getRssi(msg);
```

but `CC2420Packet` is not emulated in TOSSIM. So, this line cannot be included in a simulated program. A similar functionality is obtained with

```
RSSI=((tossim_metadata_t*) (msg
    ->metadata))->strength;
```

Both RSSI values are not equivalent. The first one is mainly a measure of transmission power and noise. The second RSSI value is a path loss between the nodes without noise. To be able to compare both of them, it is necessary to add TOSSIM, the noise to path loss. The steps undertaken were as follows: First, define the noise model in driver.c with a very low noise level.

```
for (int j=0;j<500;j++)
    m->addNoiseTraceReading( (char
      )(-100));
```

Then, load path loss inter-nodes and add Gaussian noise:

```
# define NODE(a,b,c,d) {newpow(r,a)
    ;}
# include "red.def"
```

Network topology and path loss are defined in *red.def* as an x-macro as said earlier. Function `newpow` updates the relationship between nodes and adds noise. The `newpow` code is shown in Listing 3.

Every time `beta` varies (i.e. when a message is sent), it is necessary to update network connections. An update is made following Eq. (7), as described in Section 5.

8. Results obtained

In this section, we compare the simulated results with the real experiments performed in the laboratory. These experiments consist of two devices in an indoor environment. In Fig. 4, the devices used in the measurement are shown.

```
 ...
 Mote* current = t->currentNode();
 int id = current->id();
 bool radioOn = TossimPacketModelC$running[id]; //$ transceiver status
 long long int currentTime = t->time(); // current time

 // only update battery if transceiver was on since last event on node
 if (radioStatus[id] == true) {
   long long int deltaTime = currentTime - lastEventTime[id];

   // Transceiver power based on battery status
   float power = I_RX*batteryJ2V(beta[id]);

   // battery update
   beta[id] -= (((float)deltaTime)*1e-10 * power)/MAXBATTERY;

   // avoid negative values in battery
   if (beta[id]<0) {beta[id]=0;};

   // update connectivity based on battery status
   newpow(t->radio(),id);
 }

// update mTossim status
radioStatus[id] = radioOn;
lastEventTime[id] = currentTime;
...
```

Listing 1. Structure of battery update code.

```
  float batteryJ2V(float joulesperunit)
 {
    float volts=sqrt(2*joulesperunit*MAXBATTERY/CAPACITOR);
    return volts;
 }
```

Listing 2. Battery model.

```
void newpow(Radio* r, int id)
{
#define NODE(a,b,c,d) {float decay
     = normalrandom( P_ATTR*(1-beta
    [id])+pathloss[id][a],
    VARIANCE_RSSI ); r->add(id,a,
    decay);}
#include "red.def"
}
```

Listing 3. newpow function.

To compare the results, the same program is used in both the simulator and the real telosB devices (with the exception mentioned in Section 7.2 regarding how to measure the RSSI value).

The battery used in these experiments consists of two capacitors of 70 F, 2.1 V in a serial connection. It offers a global capacity of 35 F and provides 3.3 V of power supply: the maximum operating voltage of the telosB devices.

In this situation, the energy stored by the battery can be obtained using Eq. (12).

$$E_{Batt} = \frac{1}{2} \cdot C \cdot V^2 = \frac{1}{2} \cdot 35\ \mathrm{F} \cdot (3.3\ \mathrm{V})^2 = 190\ \mathrm{J} \tag{12}$$

The minimum working voltage for telosB devices varies as a function of the characteristics of the radio transceiver and the microcontroller. In our case, its minimum work voltage is measured as 1.5 V, that is, the telosB device is going to operate in the range [3.3–1.5 V] due to the discharge of the battery through use.

Therefore, the minimum work threshold (E_{min}) for telosB can be obtained as described in Eq. (13):

$$E_{min} = \frac{1}{2} \cdot C \cdot V_{min}^2 = \frac{1}{2} \cdot 35\ \mathrm{F} \cdot (1.5\ \mathrm{V})^2 = 39\ \mathrm{J} \tag{13}$$

These values are the good ones for telosB simulations, as they use a capacitor to power the nodes. The simulator is valid for different hardware, only modifying the typical characteristics of the platform for simulation.

8.1. Battery lifetime accuracy

Several experiments that test the life-time accuracy of the simulator have been performed, changing the duty circle, that is, the relationship between the time and the radio transceiver in on state and between the time and the radio transceiver in o state. In the experiment, the device sends a message to the Base Station during every duty cycle. Fig. 5 sums up these results. Differences between the estimated voltage and the measured voltage are negligible.

These results validate the simulator for predicting the lifetime of a network, without the necessity of time-consuming and costly experiments. Moreover, they permit an evaluation of a strategic maintenance of the WSNs node before its deployment.

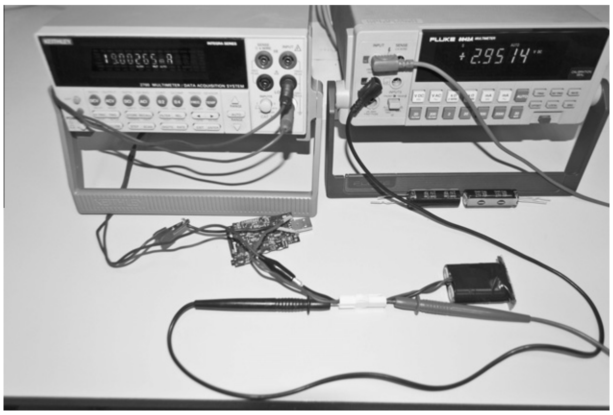
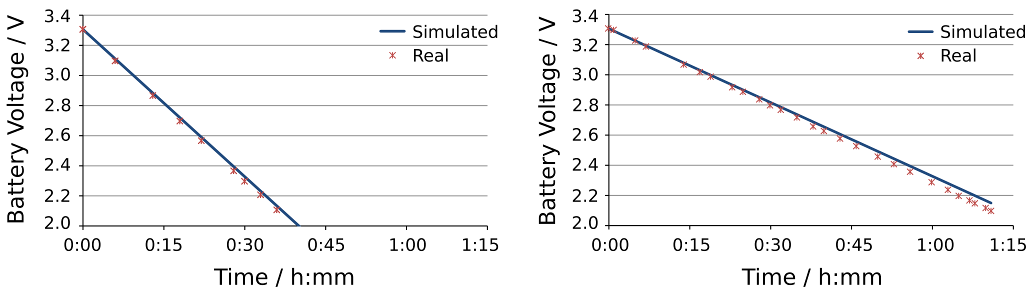


Fig. 4. Laboratory set-up for testing motes.

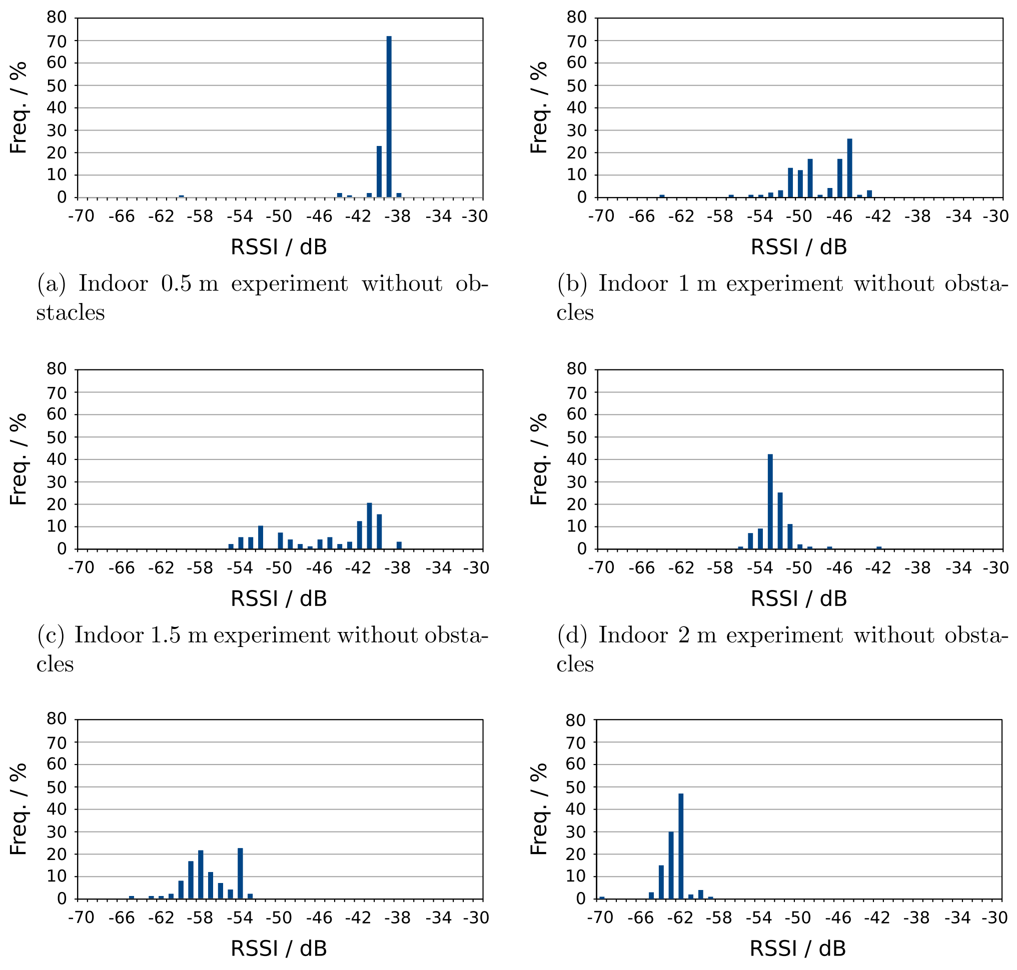


(a) Radio transceiver always on. (b) Radio transceiver 50% duty cycle.

Fig. 5. Evolution of battery voltage along time.

8.2. *RSSI estimation*

Fig. 6 depicts the histogram of the practical experiments performed in the laboratory at different distances, without obstacles between devices. Overall, 100 samples are obtained for distances in [0.5–5] m range, obtaining a new RSSI measure every second.



(a) Indoor 0.5 m experiment without obstacles

(b) Indoor 1 m experiment without obstacles

(c) Indoor 1.5 m experiment without obstacles

(d) Indoor 2 m experiment without obstacles

(e) Indoor 2.5 m experiment without obstacles

(f) Indoor 5 m experiment through one wall

Fig. 6. Indoor RSSI measurements with several inter-nodes distances.

Table 3
Resume of the information obtained with the RSSI.

Dist. (m)	Avg. RSSI (dB)	Std. deviation (dB)
0.5	−39.58	2.23
1.0	−53.12	4.03
1.5	−46.18	5.16
2.0	−52.52	1.69
2.5	−57.15	2.42

These results are summed up in Table 3, showing the average RSSI and the standard deviation of the measurement. As some models predict, RSSI tends to decrease with the distance. RSSI localization techniques are based on this characteristic.

As can be observed, the measurements have oscillations. Continuous modifications of the ambient noise and the environment produce variations in the RSSI. In general, these variations are simulated as a noise. The proposed simulator uses the Gaussian noise model, assuming a standard deviation of 5 dB, which corresponds well with the real experiments performed in our laboratory.

The simulator allows the variation of the standard deviation in order to adjust the simulation to a real environment.

Comparing the average results with the ITU model, the best approximation of the parameters are the commercial area values. Our experiments present an average *N* of 23, whereas the ITU model for a commercial area at 2.4 GHz proposes a value of 22.

The simulator has been programmed using the values of standard deviation and attenuation obtained in our test. However, to increase the accuracy, as described earlier, the models can be personalized and adjusted.

Fig. 7 depicts a comparison between the measured and simulated RSSI. In three of the five experiments, the estimated RSSI have an average error below sigma; in one, the error is below two sigma, and in the remaining ones, the errors are below 3 sigma error.

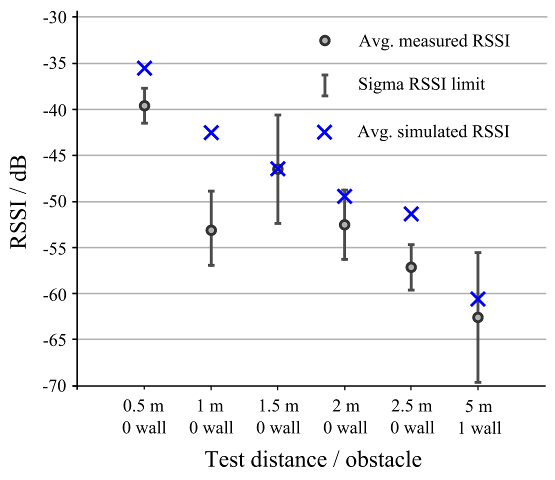


Fig. 7. Comparison between measured and simulated RSSI.

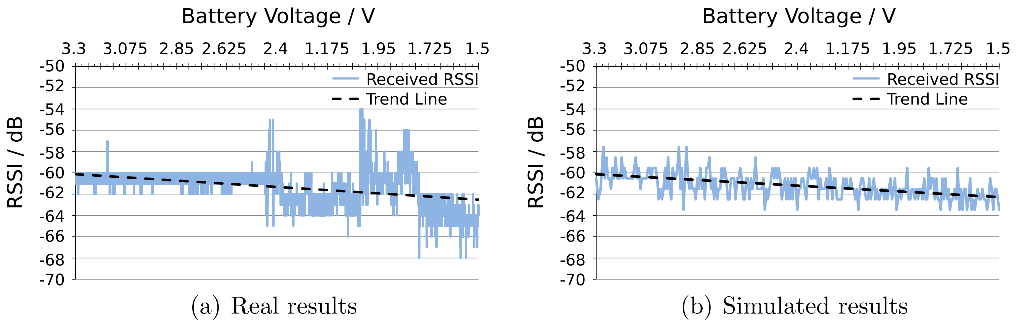


Fig. 8. Relationship between RSSI attenuation and the battery voltage using super-capacitors.

These results show that our simulator permits approximately the RSSI performance of the real experiments, considering the limits of the model. RSSI varies with reflections, but simulating reflection requires ray tracing models with an indepth knowledge of the geometry of a variable environment (location of ground planes, metallic surfaces, or elements that block RSSI transmission, such as plants, for example). However, the ITU model offers a good response and permits the prediction of the location of nodes in a scenery as a function of its coverage, taking into account this simplification of the reality.

8.3. Relationship between battery voltage and RSSI

In this experiment, the relationship between the received RSSI and the battery level has been studied. Fig. 8 shows the results obtained with a real experiment and the values obtained with the simulator. This experiment has been performed with the nodes separated by about 3 m, and considering the measured RSSI oscillations due to sporadic reflection effects.

In our indoor experiment, an average attenuation of 3 dB is obtained along the voltage range of a telosB device [3.3–1.5 V], using a capacitor of 35 F, charged at 3.3 V (the maximum operative voltage of a telosB device).

The proposed simulator takes this issue into account, and permits the selection of a power transmission attenuation according to the battery technology. In Fig. 8, an attenuation of 3 dB in the operative range is used.

9. Conclusion

In this article, an extension of the TOSSIM simulator, named *mTOSSIM*, is presented. It permits a more realistic simulation of the reduction of power transmission with the battery drop and the indoor power transmission loss. It improves TOSSIM in one of its biggest drawbacks, the too-simplified model that is used as default to evaluate the RSSI propagation.

mTOSSIM easily allows an estimation of the power loss in transmission with battery discharge and distance.

mTOSSIM permits an evaluation of a WSN of an arbitrary number of sensors. Moreover, the proposed simulator allows to obtain the lifetime of the network devices. Obtaining the lifetime with real experiments is typically a very time-consuming task.

mTOSSIM is suitable for many applications, such as localization, coverage range study, estimation of the time between network maintenance, and so on.

Moreover, it easily permits to debug nesC applications for the TinyOS operating system. To do this, it presents a graphical user interface that allows monitoring the internal registers of devices and tracing the communications between different nodes. The nesC simulated applications can be exported to real nodes by only adding minor code modifications.

The comparison between real experiments and simulations demonstrates the accuracy of the simulations, in battery-level estimation, network and node lifetime estimation, and RSSI propagation.

Currently, the authors are working on improving the predictions of the models, including the power consumption of additional hardware, such as external sensors.

Acknowledgments

This research has been supported by the Consejería de Innovación, Ciencia y Empresa, Junta de Andalucía, Spain, through the projects of excellency ARTICA (Reference Number: P07-TIC-02476) and eSAPIENS (Reference Number: TIC-5705) and by the ‘‘Cátedra de Telefónica, Inteligencia en la Red’’, Seville, Spain, through the project TORTUGA.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Computer Networks 38 (4) (2002) 393–422, http://dx.doi.org/10.1016/S1389-1286(01)00302-4.

[2] R. Aversa, B. Di Martino, M. Ficco, S. Venticinque, A simulation model for localization of pervasive objects using heterogeneous wireless networks, Simulation Modelling Practice and Theory 19 (8) (2011) 1758–1772.

[3] IEEE 802.15.4, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), September, 2006.

[4] Z. Alliance, Zigbee-2007 Specification – Revision 17, Tech. Rep., ZigBee Standards Organization, 2007.

[5] ISO/IEC 14543-3-10:2012: Information Technology – Home Electronic Systems (HES) – Part 3-10: Wireless Short-Packet (WSP) Protocol Optimized for Energy Harvesting – Architecture and Lower Layer Protocols, 2012.

[6] N. Tziritas, T. Loukopoulos, S. Lalis, P. Lampsas, Algorithms for energy-driven agent placement in wireless embedded systems with memory constraints, Simulation Modelling Practice and Theory 19 (6) (2011) 1445–1464, http://dx.doi.org/10.1016/j.simpat.2010.12.005.

[7] H. Sanson, M. Mitsuji, Localization for emergency sensor networks, in: The 7th International Conference on Advanced Communication Technology, ICACT 2005, vol. 2, Phoenix Park, 2005, pp. 982–987.

[8] A. Joshi, N. VishnuKanth I, N. Samdaria, S. Bagla, P. Ranjan, Gps-less animal tracking system, in: Proceedings of the 4th International Conference on Wireless Communication and Sensor Networks, WCSN 2008, 2008, pp. 120–125. http://dx.doi.org/10.1109/WCSN.2008.4772694.

[9] J. Galbreath, J. Frolik, Channel allocation strategies for wireless sensors statically deployed in multipath environments, in: IEEE Trans. on Intelligent Trans. Systems, vol. 2006, Nashville, TN, 2006, pp. 334–341. http://dx.doi.org/10.1145/1127777.1127828.

[10] D. Efstathiou, A. Koutsopoulos, S. Nikoletseas, Parameterized energy latency trade offs for data propagation in sensor networks, Simulation Modelling Practice and Theory 19 (10) (2011) 2226–2243, http://dx.doi.org/10.1016/j.simpat.2011.08.001.

[11] V. Ramadurai, M. Sichitiu, Localization in wireless sensor networks: a probabilistic approach, in: W. Zhuang, C. Yeh, O. Droegehorn, C. Toh, H. Arabnia (Eds.), Proceedings of the International Conference on Wireless Networks, Las Vegas, NV, 2003, pp. 275–281.

[12] J. Wu, S. Yang, Optimal movement-assisted sensor deployment and its extensions in wireless sensor networks, Simulation Modelling Practice and Theory 15 (4) (2007) 383–399, http://dx.doi.org/10.1016/j.simpat.2006.11.006.

[13] R. Bischoff, R. Wattenhofer, Analyzing connectivity-based multi-hop ad-hoc positioning, in: Proc. of IEEE Global Communications (GLOBECOM), Orlando, FL, 2004, pp. 165–174. http://dx.doi.org/10.1109/PERCOM.2004.1276855.
[14] mTossim Simulator, Freely Available in <http://grupo.us.es/ustic150/>.
[15] A. Stetsko, M. Stehlik, V. Matyas, Calibrating and comparing simulators for wireless sensor networks, Mobile Ad-Hoc and Sensor Systems, IEEE International Conference on (2011) 733–738.
[16] J. Barbancho, C. Leon, J. Molina, A. Barbancho, Sir: a new wireless sensor network routing protocol based on artificial intelligence, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 3842 LNCS (2006) 271–275. http://dx.doi.org/10.1007/11610496_35.
[17] Y.b. Wang, X.b. Wang, B.b. Xie, D.b. Wang, D.b. Agrawal, Intrusion detection in homogeneous and heterogeneous wireless sensor networks, IEEE Transactions on Mobile Computing 7 (6) (2008) 698–710, http://dx.doi.org/10.1109/TMC.2008.19 (cited by, since 1996, 46).
[18] P. Levis, N. Lee, M. Welsh, D. Culler, Tossim: accurate and scalable simulation of entire tinyos applications, in: First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), 2003.
[19] P. Levis, N. Lee, Tossim: A Simulator for Tinyos Networks, September 2003. <http://www.tinyos.net/dist-1.1.0/snapshot-1.1.11Feb2005cvs>.
[20] M. Karl, A Comparison of the Architecture of Network Simulators ns-2 and Tossim, Tech. Rep., Fakultt fr Informatik, Elektrotechnik und Informationstechnik. Institut fr Parallele und Verteilte Systeme. Abteilung Verteilte Systeme. Universitt Stuttgart, January 2005.
[21] E. Perla, A.O. Catháin, R.S. Carbajo, M. Huggard, C. Mc Goldrick, Powertossim z: realistic energy modelling for wireless sensor network environments, in: Proceedings of the 3nd ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks, PM2HW2N '08, ACM, New York, NY, USA, 2008, pp. 35–42. http://dx.doi.org/10.1145/1454630.1454636.
[22] T. Prabhakar, S. Venkatesh, M. Sujay, J. Kuri, K. Praveen, Simulation blocks for tossim-t2, in: Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on, 2008, pp. 17–23. http://dx.doi.org/10.1109/COMSWA.2008.4554371.
[23] J. Polley, D. Blazakis, J. McGee, D. Rusk, J. Baras, Atemu: a fine-grained sensor network simulator, in: Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on, 2004, pp. 145–152. http://dx.doi.org/10.1109/SAHCN.2004.1381912.
[24] M. Jung, J. Oh, Swarm-etossim: a simulator for distributed energy-constrained tiny devices, in: Autonomous Decentralized Systems (ISADS), 2011 10th International Symposium on, 2011, pp. 17–24. http://dx.doi.org/10.1109/ISADS.2011.9.
[25] N. Minar, The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations, SFI Working Papers, Santa Fe Institute, 1996.
[26] J. Polastre, R. Szewczyk, D. Culler, Telos: enabling ultra-low power wireless research, in: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, vol. 48, 2005.
[27] Chipcon Products, Cc2420. 2.4 GHz ieee 802.15.4 / zigbee-ready rf Transceiver, Tech. Rep., Texas Instruments, 2007.
[28] D. Doerffel, S. Sharkh, A critical review of using the Peukert equation for determining the remaining capacity of lead-acid and lithium-ion batteries, Journal of Power Sources 155 (2) (2006) 395–400.
[29] A. Stepanov, I. Galkin, L. Bisenieks, Implementation of supercapacitors in uninterruptible power supplies, in: Power Electronics and Applications, 2007 European Conference on, 2007, pp. 1–7. http://dx.doi.org/10.1109/EPE.2007.4417559.
[30] C. Park, K. Lahiri, Battery discharge characteristics of wireless sensor nodes: An experimental analysis, in: In Proceedings of the IEEE Conf. on Sensor and Ad-hoc Communications and Networks, SECON, 2005.
[31] C. Buratti, A. Conti, D. Dardari, R. Verdone, An overview on wireless sensor networks technology and evolution, Sensors 9 (2009) 6869–6896, http://dx.doi.org/10.3390/s90906869.
[32] D. Zuowu, W. Shulin, Z. Weijun, Q. Min, Study about lithium battery's characteristics, in: Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010 International Conference on, vol. 2, 2010, pp. 639–642. http://dx.doi.org/10.1109/CMCE.2010.5609524.
[33] G. Mao, B.D. Anderson, B. Fidan, Path loss exponent estimation for wireless sensor network localization, Computer Networks 51 (10) (2007) 2467–2483, http://dx.doi.org/10.1016/j.comnet.2006.11.007.
[34] T. Stoyanova, F. Kerasiotis, A. Prayati, G. Papadopoulos, A practical rf propagation model for wireless network sensors, in: Proceedings – 3rd International Conference on Sensor Technologies and Applications, SENSORCOMM 2009, Athens, Glyfada, 2009, pp. 194–199. http://dx.doi.org/10.1109/SENSORCOMM.2009.39.
[35] Recommendation itu-r p.1238-2. Propagation Data and Predictionmethods for the Planning of Indoor Radiocommunication Systems Andradio Local Area Networks in the Frequency Range 900 MHz to 100 GHz.itu Recommendations, 2001.
[36] T. Chrysikos, G. Georgopoulos, S. Kotsopoulos, Site-specific validation of itu indoor path loss model at 2.4 GHz, in: World of Wireless, Mobile and Multimedia Networks Workshops, 2009. WoWMoM 2009. IEEE International Symposium on a, 2009, pp. 1–6. http://dx.doi.org/10.1109/WOWMOM.2009.5282432.
[37] O. Katircioglu, H. Isel, O. Ceylan, F. Taraktas, H. Yagci, Comparing ray tracing, free space path loss and logarithmic distance path loss models in success of indoor localization with rssi, in: Telecommunications Forum (TELFOR), 2011 19th, 2011, pp. 313–316. http://dx.doi.org/10.1109/TELFOR.2011.6143552.
[38] TinyOs Documentation Wiki, Tutorial TOSSIM, October 2011. <http://docs.tinyos.net> (last visited April 2012).
[39] R. Meyers, X macros, May 2001. <http://www.drdobbs.com/cpp/184401387> (last visited April 2012).