Benchmarking Low-Resource Device Test-Beds for Real-Time Acoustic Data

Congduc Pham[1,2(✉)] and Philippe Cousin[2]

[1] LIUPPA Laboratory, University of Pau, Pau, France
congduc.pham@univ-pau.fr
[2] Easy Global Market, Madrid, Spain
philippe.cousin@eglobalmark.com

Abstract. The EAR-IT project relies on 2 test-beds to demonstrate the use of acoustic data in smart environments: the smart city SmartSantander test-bed and the smart building HobNet test-beds. In this paper, we take a benchmarking approach to qualify the various EAR-IT test-bed based on WSN and IoT nodes with IEEE 802.15.4 radio technology. We will highlight the main performance bottlenecks when it comes to support transmission of acoustic data. We will also consider audio quality and energy aspects as part of our benchmark methodology in order to provide both performance and usability indicators. Experimentations of multi-hop acoustic data transmissions on the SmartSantander test-bed will be presented and we will demonstrate that streaming acoustic data can be realized in a multi-hop manner on low-resource device infrastructures.

Keywords: Benchmark · Internet of Thing · Acoustic · Smart Cities

1 Introduction

There is a growing interest in multimedia contents for surveillance applications in order to collect richer informations from the physical environment. Capturing, processing and transmitting multimedia information with small and low-resource device infrastructures such as Wireless Sensor Networks (WSN) or so-called Internet-of-Things (IoT) is quite challenging but the outcome is worth the effort and the range of surveillance applications that can be addressed will significantly increase. The EAR-IT project (www.ear-it.eu) is one of these original projects which focuses on large-scale "real-life" experimentations of intelligent acoustics for supporting high societal value applications and delivering new innovative range of services and applications mainly targeting to smart-buildings and smart-cities. One scenario that can be demonstrated is an on-demand acoustic data streaming feature for surveillance systems and management of emergencies. Other applications such as traffic density monitoring or ambulance tracking are also envisioned and are also requiring timely multi-hop communications between low-resource nodes. The EAR-IT project relies on 2 test-beds to demonstrate the use of acoustic data in smart environments: the smart city SmartSantander test-bed and the smart building HobNet test-bed.

V.C.M. Leung et al. (Eds.): TridentCom 2014, LNICST 137, pp. 97–106, 2014.
DOI: 10.1007/978-3-319-13326-3_10

There have been studies on multimedia sensors but few of them really consider timing on realistic hardware constraints for sending/receiving flows of packets [1–7]. In this paper, we take a benchmarking approach to qualify the various test-beds based on WSN and IoT nodes with IEEE 802.15.4 radio technology. We will highlight the main performance bottlenecks when it comes to support acoustic data. We will also consider audio quality and energy aspects as part of our benchmark methodology in order to provide both performance and usability indicators. The paper is then organized as follows: Section 2 reviews the EAR-IT test-beds and especially the various sensor node hardware. We will also present some audio sampling and transmission constraints. Section 3 will present our benchmark approach and experimental results showing the main performance bottlenecks. Section 4 will present the audio hardware we developed for the IoT nodes. In Section 5 we will present experimentations of multi-hop acoustic data transmissions on the SmartSantander test-bed and an analysis of the audio quality and energy consumption of the deployed system. Conclusions will be given in Section 6.

2 The EAR-IT Test-Beds

The EAR-IT test-beds consist in (*i*) the SmartSantander test-bed and (*ii*) the HobNet test-bed. The SmartSantander test-bed is a FIRE test-bed with 3 locations. One main location being the Santander city in north of Spain with more than 5000 nodes deployed across the city. This is the site we will use when referring to the SmartSantander test-bed. The HobNet test-bed is located at MANDAT Intl which is part of the University of Geneva and it is an in-door test-bed. Many information can be found on corresponding project web site (www.smartsantander.eu and www.hobnet-project.eu) but we will present in the following paragraphs some key information that briefly present the main characteristics of the deployed nodes.

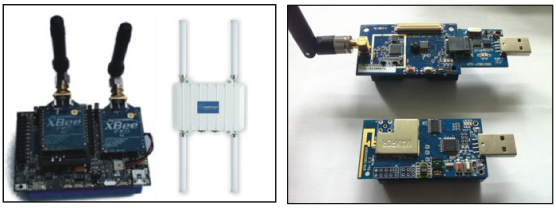

Fig. 1. Left: Santander's IoT node (left) and gateway (right). Right: HobNet's CM5000 & CM3000 AdvanticSys TelosB

2.1 The SmartSantander Test-Bed Hardware

IoT Nodes and Gateways. IoT nodes in the Santander test-bed are WaspMote sensor boards and gateways are Meshlium gateways, both from the Libelium company (www.libelium.com). Most of IoT nodes are also repeaters for multi-hops

communication to the gateway. Figure 1(left) shows on the left part the WaspMote sensor node serving as IoT node and on the right part the gateway. The WaspMote is built around an Atmel ATmega1281 micro-controller running at 8MHz. There are 2 UARTs in the WaspMote that serve various purposes, one being to connect the micro-controller to the radio modules.

Radio Module. IoT nodes have one XBee 802.15.4 module and one XBee DigiMesh module. Differences between the 802.15.4 and the DigiMesh version are that Digimesh implements a proprietary routing protocols along with more advanced coordination/node discovery functions. In this paper, we only consider acoustic data transmission/relaying using the 802.15.4 radio module as the DigiMesh interface is reserved for management and service traffic. XBee 802.15.4 offers the basic 802.15.4 [8] PHY and MAC layer service set in non-beacon mode. Santander's nodes have the "pro" version set at 10mW transmit power with an advertised transmission range in line-of-sight environment of 750m. Details on the XBee/XBee-PRO 802.15.4 modules can be found from Digi's web site (www.digi.com).

2.2 The HobNet Test-Bed Hardware

HobNet is also a FIRE test-bed that focuses on Smart Buildings. Although the HobNet test-bed has several sites, within the EAR-IT project only the UNIGE test-bed at the University of Geneva with TelosB-based motes is concerned.

IoT Nodes. Sensor nodes in the HobNet test-bed consist in AdvanticSys TelosB motes, mainly CM5000 and CM3000, see figure 1(right), that are themselves based on the TelosB architecture. These motes are built around an TI MSP430 microcontroller with an embedded Texas Instrument CC2420 802.15.4 compatible radio module. Documentation on the AdvanticSys motes can be found on AdvanticSys web site (www.advanticsys.com). AdvanticSys motes run under the TinyOS system (www.tinyos.net). The last version of TinyOS is 2.1.2 and our tests use this version.

Radio Module. The CC2420 is less versatile than the XBee module but on the other hand more control on low-level operations can be achieved. The important difference compared to the previous Libelium WaspMote is that the radio module is connected to the microcontroller through an SPI bus instead of a serial UART line which normally would allow for much faster data transfer rates. The CC2420 radio specification and documentation are described in [9].

The default TinyOS configuration use a MAC protocol that is compatible with the 802.15.4 MAC (Low Power Listening features are disabled). The default TinyOS configuration also uses ActiveMessage (AM) paradigm to communicate. As we are using heterogeneous platforms we will rather the TKN154 IEEE 802.15.4 compliant API. We verified the performances of TKN154 against the TinyOS default MAC and found them greater.

2.3 Audio Sampling and Transmission Constraints

4KHz or 8KHz periodic 8-bit audio sampling implies to handle 1 byte of raw audio data once every 250us or 125us respectively. Then, when a sufficient number of samples have been buffered, these audio data must be encoded and transmitted while still maintaining the sampling process. For instance, if we take the maximum IEEE 802.15.4 payload size, i.e. 100 bytes, the audio sample time is 25ms and 12.5ms for 4KHz and 8KHz sampling respectively. Most of IoT nodes are based on low speed microcontroller (Atmel ATmega1281 at 8MHz for the Libelium WaspMote and TI MSP430 at 16MHz for the AdvanticSys) making simultaneous raw audio sampling and transmission (even without encoding) nearly impossible when using only the mote microcontroller.

3 Benchmarking IoT Nodes

The benchmark phase is intended to determine (*i*) the network performance indicators in terms of sending latency, relay latency, relay jitter and packet loss rates, (*ii*) audio quality indicators depending on the packet loss rates and (*iii*) energy consumption indicators. Regarding the network indicators we measured on real sensor hardware and communication API the time spent in a generic send() function, noted t_{send}, and the minimum time between 2 packet generation, noted t_{pkt}. t_{pkt} will typically take into account various counter updates and data manipulation so depending on the amount of processing required to get and prepare the data, t_{pkt} can be quite greater than t_{send}. With t_{send}, we can easily derive the maximum sending throughput that can be achieved if packets could be sent back-to-back, and with t_{pkt} we can have a more realistic sending throughput. In order to measure these 2 values, we developed a traffic generator with advanced timing functionalities. Packets are sent back-to-back with a minimum of data manipulation needed to maintain some statistics (counters) and to fill-in data into packets, which is the case in a real application. On the WaspMote, we increased the default serial baud rate between the microcontroller and the radio module from 38400 to 125000. The Libelium API has also been optimized to finally cut down the sending overheads by almost 3! Figure 2 shows t_{send} and t_{pkt} as the payload is varied.

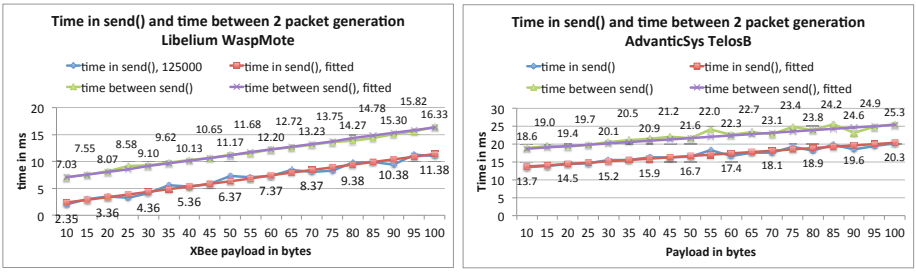

Fig. 2. Sending performances: WaspMote (left) and TelosB (right)

At the sending side, transmission of raw audio at 8KHz is clearly not possible as the time to send 100-byte packets is well above the available time window. 4KHz audio is possible on the WaspMote but not really feasible on the TelosB because the sampling process does interrupt the sending process which is already very close to the maximum time window allowed, i.e. 25ms.

In the next set of benchmark, we use a traffic generator to send packets to a receiver where we measured (i) the time needed by the mote to read the received data into user memory or application level, noted t_{read}, and (ii) the total time needed to relay a packet. Relay jitter is found to be quite small and easily handled with traditional playout buffer mechanisms.

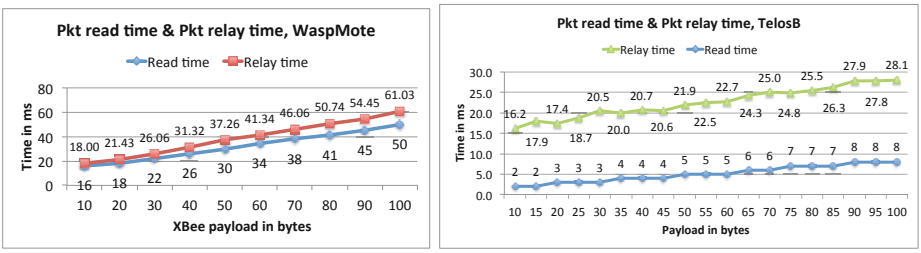

Fig. 3. Read and relaying performances: WaspMote (left) and TelosB (right)

On the WaspMote, we found that t_{read} is quite independent from the microcontroller to radio module communication baud rate because the main source of delays come from memory copies. We can see that when it comes to multi-hop transmissions, 4KHz raw audio is not feasible neither on WaspMote nor TelosB because the time window of 25ms for a 100-byte packets is not sufficient. We will describe in section 5 the audio quality and the energy benchmarking process.

4 Audio Hardware

To leverage the performance issues identified during the benchmark step, one common approach is to dedicate one of the 2 tasks to another microcontroller: (1) use another microcontroller to perform all the transmission operations (memory copies, frame formatting, ...) or (2) use another microcontroller to perform the sampling operations (generates interruptions, reads analog input, performs A/D conversion and possibly encodes the raw audio data). With the hardware platforms used in the EAR-IT project we can investigate these 2 solutions:

1. Libelium WaspMote uses an XBee radio module which has an embedded internal microcontroller that is capable of handling all the sending operations when running in so-called transparent mode (serial line replacement mode);
2. Develop a daughter audio board for the AdvanticSys TelosB mote that will perform the periodic sampling, encode the raw audio data with a given audio codec and fill in a buffer that will be periodically read by the host microcontroller, i.e. the TelosB MSP430.

Solution 1 on Libelium WaspMote. Solution 1 has been experimented and we successfully sampled at 8KHz to generate a 64000bps raw audio stream which is handled transparently by an XBee module running in transparent mode. Transmission is done very simply by writing the sample value in a register. However, we are still limited to 1-hop transmission because the transparent mode does not allow for dynamic destination address configuration making multi-hop transmissions difficult to configure. Moreover, as previously seen, the packet read overhead is very large on the WaspMote. The advantage is however to be able to increase the sampling rate from 4KHz to 8KHz when sending at 1-hop.

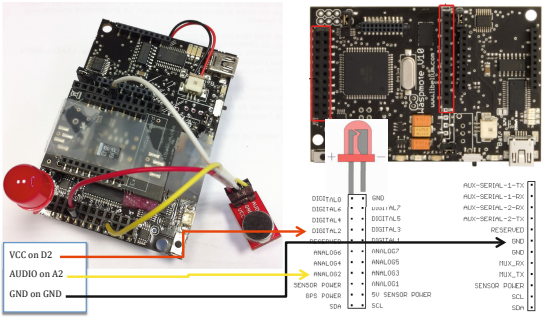

Fig. 4. Audio hardware on Libelium WaspMote

Solution 2 on AdvanticSys TelosB. The developed audio board will have its own microcontroller to handle the sampling operations and encode in real-time the raw audio data into Speex codec (www.speex.org). 8KHz sampling and 16-bits samples will be used to produce an optimized 8kbps encoded Speex audio stream (Speex encoding library is provided by Microchip). This audio board is designed and developed through a collaboration with IRISA/CAIRN research team and Feichter Electronics company (www.feichter-electronics.com). Here is a schematic of the audio board design:

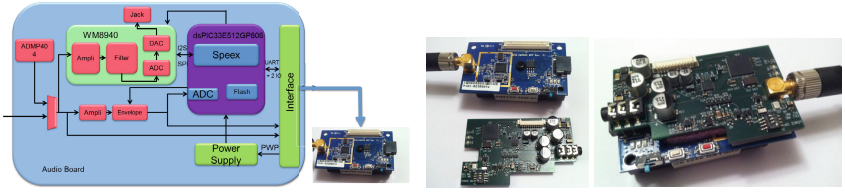

Fig. 5. Left: Audio board schematic. Right: TelosB with the audio board

The audio board has a built-in omnidirectional MEMs microphone (ADMP404 from Analog Devices) but an external microphone can also be connected. The microphone signal output is amplified, digitized and filtered with the WM8940 audio codec. The audio board is built around a 16-bit Microchip dsPIC33EP512

microcontroller clocked at 47.5 MHz that offers enough processing power to encode the audio data in real-time. From the system perspective, the audio board sends the audio encoded data stream to the host microcontroller through an UART component. The host mote will periodically read the encoded data to periodically get fixed size encoded data packets that will be transmitted wirelessly through the communication stack. The speex codec at 8kbps works with 20ms audio frames: every 20ms 160 samples of 8-bit raw audio data is sent to the speex encoder to produce a 20-byte audio packet that will be sent to the host microcontroller through an UART line. These 20 bytes will be read by the host microcontroller and 4 framing bytes are added to the audio data. The first two framing bytes will be used by the receiver to recognize an audio packet. Then sequence number can be used to detect packet losses. The last framing size stores the audio payload size (in our case it is always 20 bytes). Framing bytes are optional but highly recommended. If framing bytes are not used, only a Start Of Frame byte is inserted to allow the speex audio decoder at the receiver end to detect truncated packets.

5 Experimentations

The WaspMote mote as an audio source using solution 1 is a straightforward solution therefore the experimentations described here use the AdvanticSys TelosB mote with the developed audio board but relay nodes consist in both Libelium WaspMote and AdvanticSys motes: some TelosB motes can be used on the Santander test-bed using WaspMote as relay nodes. The receiver consists in an AdvanticSys TelosB mote connected to a Linux computer to serve as a radio gateway.

The audio source can be controlled wirelessly with 3 commands: "D" command defines the next hop address, "C" command controls the audio board power (off/on) and "A" command defines the audio frame aggregation level which will be described later on. The relay nodes can also be controlled wirelessly and they mainly accept the "D" command to define the next hop address. The receiver will get audio packets from the AdvanticSys radio gateway, check for the framing bytes and feed the speex audio decoder with the encoded audio data. The audio decoder will produce a raw audio stream that can be played in real-time with play or stored in a file by using standard Unix redirection command. A play-out buffer threshold can be specified for play to compensate for variable packet jitter at the cost of higher play-out latencies.

We selected a location in Santander near the marina, see figure 6(left), to install the audio source and the relay nodes on the same street lamps than the one deployed by the Santander test-bed, see figure 6(right). We didn't perform tests on the HobNet test-bed yet, but we use both HobNet (AdvanticSys TelosB) and Santander (Libelium WaspMote) hardware as relay nodes. We placed our nodes on the street lamps indicated in figure 6(left), at locations 11, 392, 395 and the top-right gateway. The audio node is on location 11, the receiver is at the top-right gateway location and the 2 relay nodes are at location 392 and 395. With 2 relay nodes, the number of hops is 3. Most of IoT nodes deployed in Santander can reach their associated gateway in a maximum of 3 hops. The

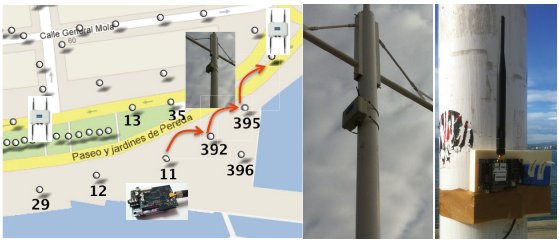

Fig. 6. Test of acoustic data streaming: topology

original IoT nodes of the Santander test-bed are placed on street lamp as shown in figure 6(left). We strapped our nodes as depicted by figure 6(right).

5.1 Multi-Hop Issues

We can see in figure 3 that on average an AdvanticSys TelosB relay node needs about 19ms to relay a 25-byte packet. However, sometimes relaying can take more than 20ms. As the audio source sends a 24-byte packet once every 20ms, it may happen that some packets are dropped at the relay node. We observed packet loss rates between 10% and 15% at the receiver. Figure 3 also shows that a WaspMote needs on average 24ms to relay a 25-byte packet. We also observed packet loss rates between 20% and 30% at the receiver.

In order to reduce the packet drop rate, we can aggregate 2 audio frames (noted A2) into 1 radio packet at the source therefore providing a 40ms time window for the relaying nodes. In this case, the radio packet payload is 48 bytes. The average relaying time is about 22ms for the TelosB and 37ms on the WaspMote as shown in figure 3. While A2 is sufficient on the TelosB to provide a packet loss rate close to 0, the WaspMote still suffers from packet loss rates between 10% and 15% at the receiver because some relaying time are greater than 40ms. On the WaspMote, we can aggregate 3 audio frames (A3) to provide a 60ms time window which is enough to relay a 72-byte packet that needs about 48ms to be relayed. Doing so succeeded in having packet loss rate close to 0.

5.2 Audio Quality Benchmarking

In order to measure the receiving audio quality, we use the ITU-T P.862 PESQ software suite for narrowband audio to get an audio quality indicator (MOS-LQO) between the original audio data and the received audio data. Figure 7 shows for various packet loss rates the MOS-LQO indicator value when there is no audio aggregation, i.e. 1 audio frame in 1 radio packet. The first vertical bar (at 4.308) is the MOS-LQO value when comparing the speex encoded audio data to the uncompressed audio format[1]. It is usually admitted that a MOS-LQO of at

[1] Reader can listen at the various audio files at web.univ-pau.fr/~cpham/SmartSantanderSample/speex

least 2.6 is of reasonably good quality. When there is a packet loss, it is possible to detect it by the gap in the sequence number and use the appropriate speex decoder mode. The red bars indicates the MOS-LQO values when packet losses are detected. Without the packet loss detection feature, missing packets are simply ignored and the speex decoder will simple decode the flow of available received packets. We can see that it is always better to detect packet losses. In figure 7 we can see that an AdvanticSys relay node without audio packet aggregation (between 10% and 15% packet loss rate) still has an acceptable MOS-LQO value. Using A2 aggregation makes the packet loss rate to be below 5% and therefore provides a good audio quality as indicated in figure 7(right). When using Libelium WaspMote as relay nodes, A3 aggregation with packet losses detection gives a MOS-LQO indicator of 3.4 and 2.9 for 5% and 10% packet loss rates respectively.

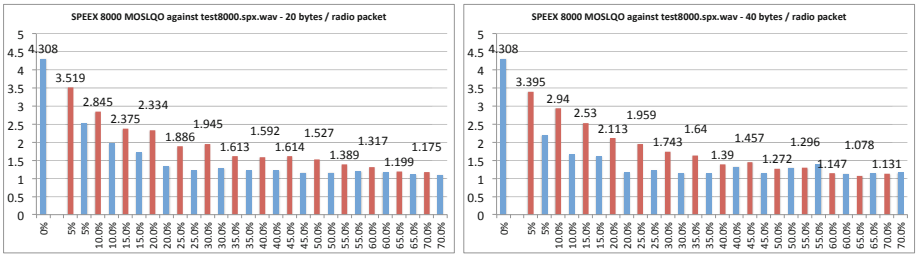

Fig. 7. MOS-LQO: A1(left) and A2(right) aggregation when pkt loss rate is varied

5.3 Energy Consumption Benchmarking

We also investigated the energy consumption of the audio source TelosB node with the developed audio board. Figure 8(left) plots the measured energy consumption every 20ms. The first part of the figure shows the idle period where the audio board is powered off and the radio module is on. Then, starting at time 43s, the audio board is powered on to capture and encode in real-time during about 20s. The audio packets are sent wirelessly. Figure 8(right) shows the cumulated energy consumption.

During idle period, the consumed energy is about 0.068J/s (68mW). During audio capture with the radio sending, the consumed energy is about 0.33J/s

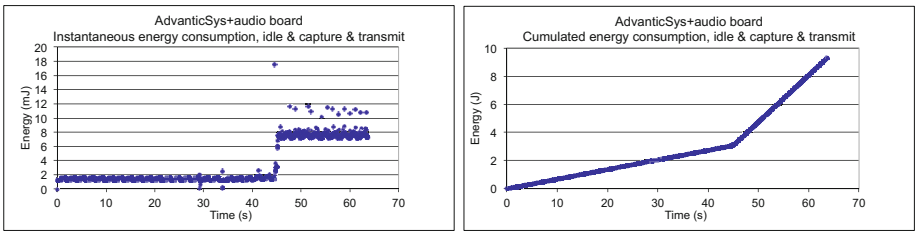

Fig. 8. Instantaneous (left) and Cumulated (right) energy consumption

(330mW). With 2 AA batteries providing about 18700J, we could continuously capture and transmit during more than 15 hours (2700000 audio frames)! With the WaspMote (although not shown due to space limitation), the 8KHz sampling and transmission process consumed about 0.610J/s (610mw) giving a continuous capture during more than 8 hours.

For relay, the WaspMote relay consumed about 0.236J/s (236mW) in listening mode and 0.238J for relaying a 72-byte radio packet in A3 mode, 3 audio frames (60B) + 3*4 framing bytes (12B). Again, with 2 AA batteries, in the best case the relay node can relay about 78606 radio packets before energy is down, i.e. 1h20m of audio. Data transmission in relaying has to use the API mode therefore the energy consumption is higher than in the case of transparent mode. However, given the results of our benchmarking process, we believe that periodic audio streaming scenarios are very possible in the context of a smart cities where most of sensor nodes can usually be recharged at night.

6 Conclusions

We took a benchmarking approach to study how acoustic data can be handled on low-resource device test-beds, highlighting communication overheads and bottlenecks that dramatically limit the relaying operations. We developed an audio board to sample and encode in real-time acoustic data and presented experimentations on the Santander EAR-IT test-bed for real-time acoustic data streaming. With appropriate audio aggregation to fit into relaying capabilities we demonstrated that streaming acoustic data is feasible on Smart Cities infrastructures with reasonably high audio quality and node lifetime.

Acknowledgments. This work is supported by the EU FP7 EAR-IT project, http://www.ear-it.eu

References

1. Rahimi, M., et al.: Cyclops: In situ image sensing and interpretation in wireless sensor networks. In: ACM SenSys (2005)
2. Mangharam, R., Rowe, A., Rajkumar, R., Suzuki, R.: Voice over sensor networks. In: 27th IEEE International of Real-Time Systems Symposium (2006)
3. Luo, L., et al.: Enviromic: Towards cooperative storage and retrieval in audio sensor networks. In: IEEE ICDCS (2007)
4. Misra, S., Reisslein, M.: A survey of multimedia streaming in wireless sensor networks. IEEE Communications Surveys & Tutorials **10**, 1174–1179 (2008)
5. Brunelli, D., Maggiorotti, M., Benini, L., Bellifemine, F.L.: Analysis of audio streaming capability of zigbee networks. In: Verdone, R. (ed.) EWSN 2008. LNCS, vol. 4913, pp. 189–204. Springer, Heidelberg (2008)
6. Turkes, O., Baydere, S.: Voice quality analysis in wireless multimedia sensor networks: An experimental study. In: Proceedings of ISSNIP (2011)
7. Touloupis, E., Meliones, A., Apostolacos, S.: Implementation and evaluation of a voice codec for zigbee. In: IEEE ISCC (June 2011)
8. IEEE, Ieee std 802.15.4-2006. (2006)
9. Texas Instrument (accessed 4/12/2013). http://www.ti.com/lit/gpn/cc2420