A Practical Solution for Time Synchronization in Wireless Sensor Networks

Eugen COCA, Valentin POPA
Stefan cel Mare University of Suceava, 720229, Romania
eugen.coca@usv.ro

***Abstract*—Time synchronization in wireless sensor node networks is a hot topic. Many papers present various software algorithms and hardware solutions to keep accurate time information on mobile nodes. In terms of real life applications wireless sensor nodes are preferred in many domains, starting with simple room monitoring and finishing with pipeline surveillance projects. Positioning applications are far more restrictive on timekeeping accuracy, as for the velocity of nodes calculations precise time or time difference values are needed. The accuracy of time information on nodes has to be always correlated with the application requirements. In this paper, we present some considerations regarding time synchronization linked with specific needs for individual practical applications. A practical low energy method of time keeping at node level is proposed and tested. The performances of the proposed solution in terms of short and long term stability and energy requirements are analyzed and compared with existing solutions. Simulation and experimental results, some advantages and disadvantages of the method are presented at the end of the paper.**

***Index Terms*—time synchronization, wireless sensor node, network protocol, lifetime estimation, clock drift**

I. Introduction

The main idea when developing a sensor network is to obtain a low-power network of nodes, each of them integrating a microcontroller, a number of local sensors and an energy source or even mini-actuators [1-2]. These sensor networks have the ability to measure different electrical and non-electrical parameters and may be used to monitor the integrity of structures in real time, to monitor pipelines, to measure the local temperature, humidity, luminosity, to detect the position and eventually the velocity of each individual mobile node.

High precision synchronization algorithms are reported with as-low-as micro-second precision [3-10], but only in few practical situations this kind of accuracy is really needed. Even with the time theoretically synchronized so precisely, there are other factors influencing the overall behavior of each individual node or the whole system. Considering only the clock drift due to temperature variations or crystal aging, radio signals propagation delay and processing time on node microcontroller level, the overall time synchronization accuracy decreases dramatically.

In time dependant applications, no time steps are allowed as events have to be recorded in an ascendant order. The same is implemented in the NTP - Network Time Protocol [11], the protocol used to synchronize the clock of computers. If the difference between the time information received by the node from the reference node and the local clock is greater than a certain value, the local node clock needs to be adjusted by modifying the reference clock frequency, like in the Network Time Protocol (NTP) [11]. This operation is time, processing power and energy consuming. When a local Real Time Clock (RTC) is involved, there is no need for time steps if the local clock drift is low enough and/or the re-synchronizations are made frequently.

This paper was supported by the project "Progress and development through post-doctoral research and innovation in engineering and applied sciences - PRiDE - Contract no. POSDRU/89/1.5/S/57083", project co-funded from European Social Fund through Sectorial Operational Program Human Resources 2007-2013.

Digital Object Identifier 10.4316/AECE.2012.04009

II. Time Synchronization Algorithms from the Energy Consuming Perspective

Many time synchronization software algorithms were developed during the last years, most of them derived from the NTP algorithm used in computer networks, while others were developed from the beginning. Despite the diversity of time keeping solutions, in the vast majority of applications, the algorithms need to run on wireless sensor nodes individually powered by their own batteries. One may also take into consideration applications involving such wireless nodes, energy being the most restrictive resource. We do not focus on applications having their nodes supplied from the mains or from a wired power supply system, as the time synchronization solutions in such cases are not relevant as the time information may be transmitted through the wire, without any radio communication. Even if the radio communication is used to collect data as well as for time synchronization, the energy consumed by the nodes does not affect their lifetime and is not a critical design requirement.

In this paragraph, we present in brief the main algorithms, focusing mostly on the synchronization precision and energy requirements instead on the working mechanism itself [3-33].

The Reference Broadcast Synchronization (RBS) algorithm uses a scheme in which each node sends reference beacons to their neighbors, using physical-layer broadcasts. A reference broadcast does not contain explicit timestamp information. Receivers use its arrival time as a reference for comparing their own clocks. RBS can be used without any external time reference, forming a relative local timescale or can maintain microsecond-level synchronization to an external one, such as UTC, if the central node is synchronized with an external source, like a GPS or an NTP time server. The real time may be kept on a central control node, or on a computer running the core application

software.

The main limitation of the RBS algorithm is it requires a network with a separate physical broadcast channel. It cannot be implemented in a point to point network. However, it is applicable for a wide range of applications in both wired and wireless networks where a broadcast domain exists and higher-precision or lower energy synchronization is required. If both high precision and low energy are required simultaneously, RBS is not the best solution, at least in battery powered applications.

The Timing-sync Protocol for Sensor Networks (TPSN) has been developed to overcome the limitations of the RBS. In TPSN the wireless sensor nodes are arranged in a virtual tree and a root node is designated. Starting from the root node to the most distant node the time synchronization procedure is performed by exchanging the time messages. The precision is two times higher than the RBS due to the two-way exchanged messages and the MAC layer time stamping. If for some reasons the root node is unavailable (out of range due to difficult propagation conditions, power failure, hardware or software failures, etc.), a new root node election process is started and a new tree is rebuilt. In terms of energy, TPSN is not recommended for highly mobile sensors, as at each tree rebuilt each node consumes extra power for resynchronization.

Single broadcast time-stamped messages are used to synchronize the time of the receiving nodes in Flooding Time Synchronization Protocol (FTSP). Synchronization errors and clock drift at wireless sensor node level are minimized using MAC layer timestamp and linear regression, respectively. There is no root node, as in multi-hop networks FTSP dynamically elects the root node. The elected root node maintains the reference time, while all other nodes will be synchronized using this information. Energy requirements are higher at the root node compared to the rest of nodes, thus generating different lifetimes of the batteries.

Greunen and Rabaey [3] proposed the Lightweight Tree-based Synchronization (LTS) focusing on simplifying the synchronization algorithm complexity rather than maximizing the accuracy. The main idea is the required maximum time synchronization accuracy in wireless sensor network is relatively low, so there is no need to have ultra-precise time information on node level. A spanning tree needs to be constructed initially, pairwise synchronization being done along the n-1 edges of the tree. As clock drift at node level is generally constant, the reference node calculates the time interval the single synchronization step is valid. In this topology, the tree depth should be minimized, as it affects the time information on nodes on edges. As keeping the time at node level requires the microcontroller to keep running all the time (even in low power mode), the energy requirements are high enough for this protocol not to be very effective in low power applications. Also, due to different clock drifts on different node (due to temperature variations for example), the time between two consecutive synchronizations may need to be reduced, thus affecting the lifetime of the nodes.

Complex time synchronization and time keeping algorithms require complex hardware and software solutions, and is high energy demanding. Wireless sensor nodes are not designed for time keeping. If the accuracy of the time is not under several milliseconds, instead of synchronizing the time every time the network topology changes it may be a good solution to get the information from a local separately powered clock.

In order to keep the local clock running on nodes in a wireless sensor network, the microcontrollers must be in the active run mode during the intervals the time information is required. During stand-by, sleep and deep-sleep modes there is no time information available on the node. Even the synchronization algorithms are able to assure the synchronization of the local clock by using the information transmitted by neighbor nodes, this also uses radio communications. If the processor is in the active running mode all the time, the energy requirements are too high and the "low-power" constrain for the node is not fulfilled. Current consumption in microcontrollers increases directly with the clock frequency, so the application designers try to keep this frequency as low as possible, in order to keep the consumption down. These techniques are not compatible with applications where time synchronization is mandatory.

Enough powerful energy resources are necessary in order for a wireless sensor node to keep a good local time. Solutions with GPS receivers, local NTP time servers, OCXO (Oven-Controlled Crystal Oscillator) local oscillators and other similar ones are not to be considered in any practical situations, due to their high-energy demands.

For a GPS receiver to work on such a node, at least 0.5 to 5 Wh of energy is required. A regular battery pack has at most 8 to 16 Wh, which means the node is able to run no more than 3 to 32 hours with the GPS module in running state (we have not to forget the other components are also running at the same time, including the microcontroller). Even if the GPS receiver is activated only just before transmitting the data (in order to put real timestamp marks on events), the necessary running time of the GPS receiver is about 3 to 5 minutes (for hot-sync), thus reducing the battery life. A good local NTP server also implies costs and energy consuming components. An OCXO time source could not be seriously considered in any low-power battery-powered application.

If the nodes are powered from the mains - due to the reasons presented above, the wireless communication may not be necessary. There are available, for example, power line communication solutions at very low prices. If power cords connect to each individual node, the wireless transmission may be not necessary and not always desirable.

III. Local Clock Stability on Wireless Sensor Nodes

In terms of application requirements, both short term and long term stability of the local RTC on nodes are important. For short term stability - minutes, even tens of minutes, when the microcontroller is powered continuously, there is really no need to resynchronize the local node clock too frequently, as for small time intervals the crystal oscillator drift is low enough in order to assure accurate time [34-36]. When taking into consideration long term stability - days, even weeks, we do not have to count on microcontroller's crystal stability, as the energy requirements are too high.

Both for short term and long term situations, the local RTC (including a regular crystal) is a good solution in terms

of precision and energy requirements - Table I.

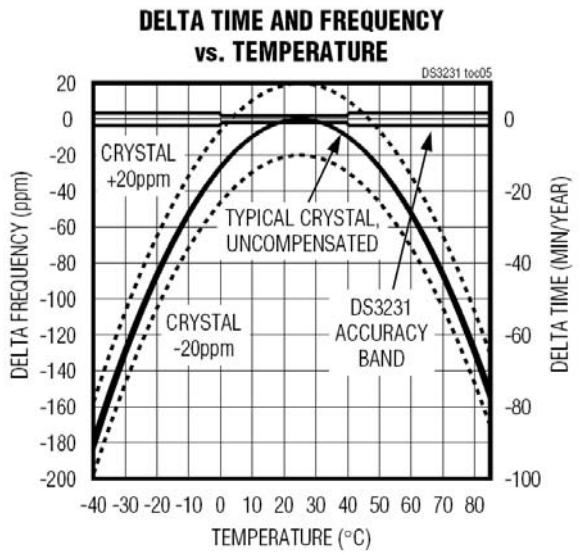

Figure 1. Crystal frequency vs. temperature. [14 si DS3231 datasheet]

If a higher precision is required, the crystal may be temperature compensated (TCXO), with drift as low as 1-2 ppm for the entire range of working temperature, from -40 to +85 degrees [15]. A summary of the energy requirements versus time synchronization accuracy is presented in Table I.

TABLE I. ENERGY REQUIREMENTS VERSUS TIME SYNCHRONIZATION ACCURACY

Energy / Accuracy	Microcontroller crystal	Radio synchronization protocols	Local RTC circuit
Short term	Low / High	Medium / High	Low / High
Long term	High / Low	High / High	Low / High

IV. LOCAL REAL TIME CLOCK AS A HIGH PRECISION AND LOW ENERGY SOLUTION

Our scenario includes a network of sensors, and each of them has local sensors measuring the temperature. One of the main requirements, at regular intervals - usually from 1 to 10 minutes, the measured values need to be time stamped and recorded in node's local non-volatile memory. It is imperative for the time stamping information to be as accurate as possible, preferably with less than 100 milliseconds. Periodically, at intervals varying from 12 to 48 hours, the recorded values are transmitted from the nodes to the central coordinator.

The above conditions may be achieved by using a network of regular battery-powered wireless sensor nodes. Time synchronization may be done via wireless using one of the known protocols.

Regarding the time-stamping of measured data there are two possibilities: one of them is to keep the clock running at each node level all the time and the other is to synchronize the time every time a measurement is made. In the first case, the energy requirements are probably high enough for the method not to be adequate due to high energy consumption, without taking into consideration the drift of the local clock.

In order to overcome these limitations we propose a practical method to synchronize the clock of each node by using a local RTC (Real Time Clock) circuit. Periodically, the time is synchronized over the wireless interface, but the time information is maintained locally, with very low energy requirements. The novelty of the proposed solution is linked to the synchronization strategy. By knowing the precision requirements of the time-stamping process we adapt the interval between two synchronizations over the wireless interface in order to be at the maximum possible value.

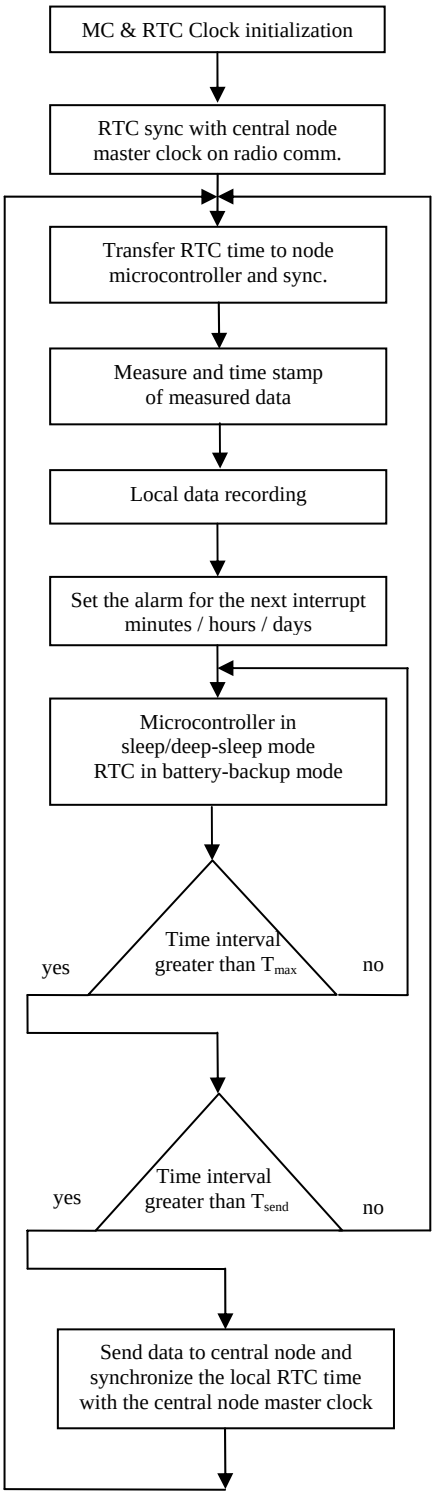

Figure 2. Block diagram of the proposed time synchronization method using a local real time circuit on every wireless sensor node

By taking into account the drift of local RTC clock it is

possible to obtain the desired accuracy using local time information only for periods from several hours to several days. After the initialization phase of the microcontroller, the node local clock is synchronized with the central node clock. A first measurement is made and recorded locally. The alarm on the node is set for the next wake-up time (current time plus T_{max}). During the interval between two consecutive temperature measurements the microcontroller is in sleep / deep sleep mode (depending on the hardware architecture) while the RTC is running in battery backup mode. When the time for alarm arrives, the RTC interrupt line will be activated. Periodically (T_{send}), the measured values which are stored locally in the non volatile microcontroller's memory are transferred to the central node, in parallel with a time synchronization of node local RTC. In multiple node network this interval may be set to different values in order to avoid collisions on the wireless interface.

V. Experimental Validation of the Proposed Time Keeping Low Energy Solution

The hardware used for performance testing was build around an ATMEGA168 microcontroller on an Arduino platform and a DS3231SN RTC circuit (Fig. 3). The RTC was battery powered in stand-by mode (on pin 14) by a 3.0V CR2032 Lithium cell with a nominal capacity of 225mAh (Fig. 4).

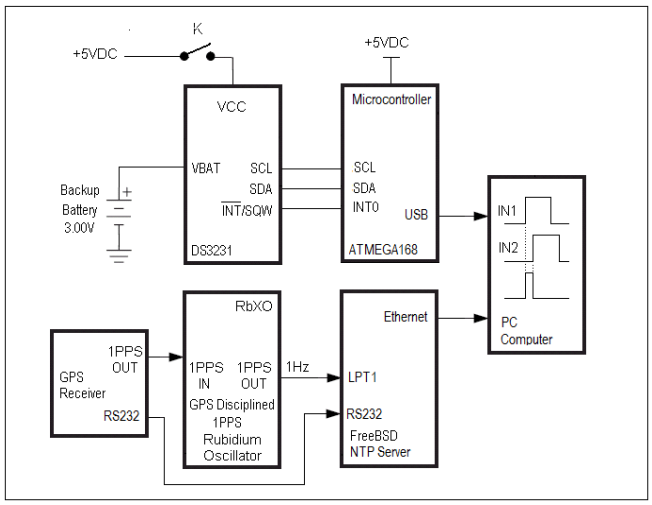

Figure 3. Block diagram of the system used for comparing the stability of the RTC clock with the stability of a GPS disciplined Rubidium Oscillator (RbXO)

The measured consumption of the RTC was about 2.1µA at 19 degrees Celsius, thus determining a backup-battery lifetime of more than 10 years (considering also the leakage current when the main power supply is present, about 100nA). In running mode the power was assured by the 5V power supply from the Arduino platform or, alternatively, by a pair of 1.5V cells, without affecting the generality of the proposed solution. The RTC circuit is powered only for the moments the measurements are made (up to 10 seconds), the rest of the time only the backup battery supply is active (5 minutes up to several hours).

The node microcontroller was programmed to work in power-down mode [38] (with oscillator off and all other internal blocks disabled) waiting for the next interrupt to appear. In this mode the consumption is only about 100nA@1.8V. The interrupt pin of the microcontroller is connected to the output interrupt pin of the RTC circuit which is programmed, after a measurement cycle, to generate an alarm.

In Fig. 3 it is shown also the diagram of the circuit used to validate the proposed solution. A PC Computer runs the software used to compare the differences between the time information delivered by the microcontroller and the real time. The PC is synchronized with the NTP (Network Time Protocol) from a local source, a time server running FreeBSD operating system, a GPS receiver and a GPS disciplined Rubidium Oscillator (RbXO) 1PPS (Pulse Per Second) time source. By measuring the difference between the time delivered by the microcontroller and the PC time one may obtain quantitative information about the precision of the proposed solution.

The RTC circuit has a time granularity of 1 second. This may be unacceptable for applications with low duty-cycles (for example the transmission interval of the measured values from sensors are five minutes) [38]. But we have to remember the drift is very low, so there is no need so synchronize with the master node every time a transmission is made.

The proposed scheme for time synchronization for a node with local RTC circuit is presented below in Fig. 4.

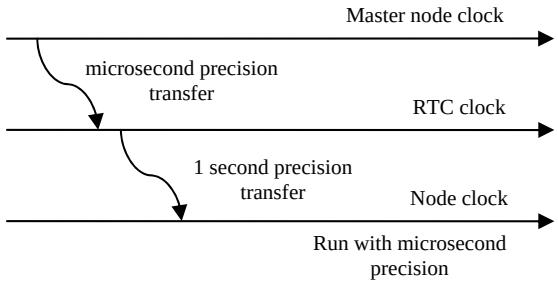

Figure 4. Time transfer operations between the master node and the node clock

All the synchronization operations on the radio interface are done only if the time difference between the last sync and the current time is greater than a reference value, determined by experiments. For example, if the precision of the time stamp on each measurement is several milliseconds, T_{max} may be of several hours, days to more than a month. Even for higher precisions, if the time from the RTC is transferred to the microcontroller clock, the synchronization precision between the time on the microcontroller node and the master clock is better than 1 second as declared in [38]. By enabling the square wave 1Hz output of the RTC circuit will bring the opportunity to use the signal fronts to better sync the time on the microcontroller.

Even the granularity of the RTC clock is 1 second the synchronization operation with the master node clock is made with microsecond precision or at the best possible precision given by the software synchronization algorithm implemented in the respective wireless sensor node network.

As a comparison, this means even if the clock is displayed with only seconds, the precision of the respective clock may be better than that. The same situation is present in the computer clocks, where the time is displayed with only

seconds but the difference between the local time and the time server may be of only several microseconds.

In order to be able to compare the lifetime of a battery-powered wireless sensor node in both cases, the first one with the time synchronized by using a radio protocol and the second one when the time is provided by the local clock with a minimum of radio communication activity, we need to know the exact energy consumption of the microcontroller running the time synchronization algorithm. We may ignore the energy consumption of the sensors and for the execution of other software routines because they are time synchronization independent activities.

By analyzing the clock synchronization algorithms used in wireless sensor networks, one may conclude all of them imply frequent radio transmissions operations, the most energy consuming job on every wireless sensor node, regardless of their structure, implementation, operating system, etc. For example, when synchronizing the clock over the wireless connection, the current consumption of a CC2431 from the battery is about 17mA / 19mA for Tx/Rx modes respectively [14], well above the maximum of 300 µA (@5.5VDC) consumed by the DS3231 RTC chip in active running mode [37].

Regarding the operating conditions, the operating temperature interval of DS3231SN is from -45 ºC to +85 ºC, so all kinds of applications are well covered, without the necessity to thermal conditioning the whole electronic module due to this device. The typical operating circuit for DS3231 is shown below [37].

VI. Advantages of the Proposed Solution

In order to justify the utility of the proposed time synchronization method, a few advantages - compared with the previous existing solutions, have to be presented:

1. No radio communication with nodes is necessary when a new measurement is done at the node level because the time stamping information is provided locally, by the RTC chip. This may be done in large intervals, depending on the measurement duty cycle and the RTC clock drift.

2. The power requirements of the proposed solution are much lower due to low power consumption of the local RTC chip compared with the consumption of the communication circuitry.

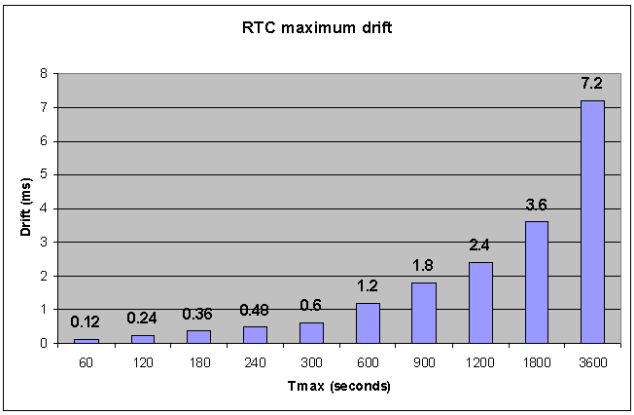

Figure 5. Maximum drift of the DS3231 RTC if the accuracy of the oscillator is 2ppm

3. In our experiments, the local RTC is a better solution if the time interval between two consecutive measurements is between 1 and 60 minutes with an estimated accuracy better than 10ms (Fig. 5). In such a case, the maximum drift of the RTC will be about 7ms.

4. Radio frequency spectrum usage is radically reduced, as only a few percent of the total communication time is used in this solution. Also, by carefully choosing a flood synchronization algorithm between the central node and the mobile nodes, the energy requirements are drastically reduced.

5. The proposed synchronization method has a low degree of vulnerability against attacks, as the time received from the central node may be always compared with the local clock, and in case of differences greater than a certain predefined value, the information received on the radio interface may be neglected. For extremely time critical environments other security checks have to be considered.

6. The proposed solution may be used on all kinds of sensor nodes, no matter the microcontroller is, including those measuring humidity, pressure, voltage, and other electrical and non-electrical quantities.

VII. Conclusion

We proposed a simple method of keeping the time accurate on a wireless sensor node network, by adding a low-cost and low-power circuit to every node. The time is kept locally, without the need to permanently synchronize the time on the node before every data transmission. This software eases the design of nodes, adding more flexibility to the application designer. The saved energy on the node is available for the main task, instead of consuming it on bulky timekeeping algorithms. The precision of the time stamp on every event depend only on the accuracy of the local lock circuit, essentially on the quality of the crystal oscillator.

This solution has to be seriously considered for real applications due to it's' effectiveness and low lost.

References

[1] F. Sivrikaya, B. Yener, "Time Synchronization in Sensor Networks: A Survey", Network, IEEE 2004, doi: 10.1109/MNET.2004.1316761.

[2] V. Shnayder, C. Borrong, L. Konrad, R. F. Thaddeus, J. Fulford, M. Welsh, "Sensor Networks for Medical Care", Technical Report TR-08-05, Division of Engineering and Applied Sciences, Harvard University, 2005.

[3] J. V. Greunen, J. Rabaey, "Lightweight Time Synchronization for Sensor Networks", Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA), San Diego, CA, September 2003, doi: 10.1145/941350.941353.

[4] J. Elson, L. Girod, D. Estrin, "Fine-grained network time synchronization using reference broadcasts", In Proc. Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, December 2002, doi: 10.1145/844128.844143

[5] M. Maroti, B. Kusy, G. Simon, A. Ledeczi, "The flooding time synchronization protocol", In Proc. SenSys'04, Baltimore, MD, November 2004, doi: 10.1145/1031495.1031501

[6] J. Elson, D. Estrin, "Time Synchronization for Wireless Sensor Networks", International Parallel and Distributed Processing Symposium (IPDPS 2001), Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, San Francisco, USA, April 2001.

[7] J. Elson, K. Raomer, "Wireless Sensor Networks: A New Regime For Time Synchronization", In Proceedings of the First Workshop on Hot Topics In Networks (HotNets-I), Princeton, New Jersey, October 2002.

[8] I. Shames, A. N. Bishop, "Relative Clock Synchronization in Wireless Networks", IEEE Communications Letters, Vol. 14, No. 4, 2010, doi: 10.1109/LCOMM.2010.04.092008.

[9] C. Benzaid, A. Saiah, N. Badache, "Secure pairwise broadcast time synchronization in wireless sensor networks", International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011, doi: 10.1109/DCOSS.2011.5982217.

[10] R. Fengyuan, L. Chuang, L. Feng, "Self-Correcting Time Synchronization Using Reference Broadcast In Wireless Sensor Network", IEEE Wireless Communications, August 2008, doi: 10.1109/MWC.2008.4599225.

[11] D. L. Mills, "Computer Network Time Synchronization: The Network Time Protocol", CRC Press, 2009, ISBN: 978-0849358050

[12] J. Koo, R. K. Panta, S. Bagchi, L. Montestruque. "A Tale of Two Synchronizing Clocks", In Proc. of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09), pp. 239-252, Nov. 2009, doi: 10.1145/1644038.1644062.

[13] F, Agyei-Ntim, K. E, Newman, "Lifetime Estimation of Wireless Body Area Sensor Networks Using Probabilistic Analysis", Wireless Personal Communications, ISSN 1572-834X, doi: 10.1007/s11277-012-0548-z.

[14] CodeBlue: Wireless Sensors for Medical Care, Available: http://fiji.eecs.harvard.edu/CodeBlue.

[15] D., Malan, T., Fulford-Jones, M., Welsh, S., Moulton, B., CodeBlue: An ad hoc sensor network infrastructure for emergency medical care, in Proc. MobiSys / Workshop Appl. Mobile Embedded Syst, Jun. 2004, pp. 12–14.

[16] I. Fernandez, A. Asensio, I. Gutierrez, J. Garcia, I. Rebollo, J. De No, "Study of the communication distance of a MEMS Pressure Sensor Integrated in a RFID Passive Tag," Advances in Electrical and Computer Engineering, vol. 12, no. 1, pp. 15-18, 2012. doi:10.4316/AECE.2012.01003

[17] A, Sobeih, J. C, Hou, "A simulation framework for sensor networks in J-Sim", Technical Report UIUCDCS-R-2003-2386, 2003.

[18] Chipcon, "CC2420 Zigbee/IEEE 802.15.4 RF Transceiver", Available: www.ti.com.

[19] E, Belding-Royer, C, Perkins, "Evolution and future directions of the ad hoc on-demand distance-vector routing protocol", Ad Hoc Networks Journal, 1(1), 125–150, 2003, doi: 10.1016/S1570-8705(03)00016-7.

[20] E. E, Egbogah, A. O, Fapojuwo, "A survey of system architecture requirements for health care-based wireless sensor networks", vol. 11, no. 5, pp. 4875-49898, Sensors, 2011, doi:10.3390/s110504875.

[21] S, Cui, R, Madan, A. J, Goldsmith, S, Lall, "Cross-layer energy and delay optimization in small-scale sensor networks", IEEE Trans. Wirel. Commun. 2007, 6, 3688–3699, 2007, doi: 10.1109/TWC.2007.060072.

[22] L, Shi, A. O, Fapojuwo, "TDMA scheduling with optimized energy efficiency and minimum delay in clustered wireless sensor networks", IEEE Trans. Mob. Comput. 2010, 9, 927–940, doi: 10.1109/TMC.2010.42.

[23] Y, Wu, P. A, Chou, S. Y, Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding", IEEE Trans. Commun. 2005, 53, 1906–1918.

[24] P. K, Loh, H. W, Jing, Y, Pan, "Performance evaluation of efficient and reliable routing protocols for fixed-power sensor networks", IEEE Trans. Wirel. Commun, 2009, 8, 2328–2335.

[25] O., Younis, S., Fahmy, "Distributed Clustering in Ad-Hoc Sensor Networks: A Hybrid, Energy-Efficient Approach", In Proceedings of the The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, 7–11 March 2004, pp. 629–640.

[26] A., Rowe, R., Mangharam, R, Rajkumar, "RT-Link: A Time-Synchronized Link Protocol for Energy-Constrained Multi-hop Wireless Networks", 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006, doi:10.1109/SAHCN.2006.288496

[27] Anthony Rowe, Rahul Mangharam, and Raj Rajkumar, "RTLink: A Time-Synchronized Link Protocol for Energy-Constrained Multi-hop Wireless Networks", CMU Tech Report TR05-08, 2005.

[28] A. Eswaran, A. Rowe, R. Rajkumar, "Nano-RK: an Energy-aware Resource-centric RTOS for Sensor Networks", IEEE Real-Time Systems Symposium, 2005, doi: 10.1109/RTSS.2005.30.

[29] J. Polastre, J. Hill, D. Culler, "Versatile low power media access for wireless sensor networks", SenSys, November 2005, doi: 10.1145/1031495.1031508.

[30] W. Ye, J. Heidemann, D. Estrin, "An energy-efficient mac protocol for wireless sensor networks", INFOCOM, June 2002, doi: 10.1109/INFCOM.2002.1019408.

[31] T. Dam, K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks", SenSys, November 2003, doi: 10.1145/958491.958512.

[32] A. El-Hoiydi, J. Decotignie, "Wisemac: An ultra low power mac protocol for the downlink of infrastructure wireless sensor networks", ISCC, 2004, doi:10.1109/ISCC.2004.1358412.

[33] V. Rajendran, K. Obraczka, J. J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks", Sensys, 2003, doi:10.1145/958491.958513

[34] ***, MAXIM DS1307 64 x 8, Serial, I2C Real-Time Clock DataSheet, 2008.

[35] ***, MAXIM DS1338 I2C RTC with 56-Byte NV RAM DataSheet, 2012.

[36] ***, APPLICATION NOTE 58, "Crystal Considerations with Maxim Real-Time Clocks (RTCs)", 2001.

[37] ***, MAXIM DS3231 Extremely Accurate I2C-Integrated RTC/TCXO/Crystal, 2010.

[38] ***, 8-bit Atmel Microcontroller with 4/8/16K Bytes In-System Programmable Flash Datasheet, ATMEL, 05/2011

\