Improvement of Energy Consumption for "Over-The-Air" Reprogramming in Wireless Sensor Networks

Konstantin Mikhaylov[#1] and Jouni Tervonen[#2]

[#] *RFMedia Laboratory, Oulu Southern Institute, University of Oulu*

Vierimaantie 5, 84100, Ylivieska, Finland

[1] konstantin.mikhaylov@oulu.fi

[2] jouni.tervonen@oulu.fi

Abstract—**New hardware platform architecture for "over-the-air" reprogramming in Wireless Sensor Networks (WSN) is presented in this article. The suggested approach is based on the usage of microcontroller RAM memory for running the program while the program memory is being changed. Implementation of suggested solution does not require any additional components and is applicable to a wide range of microcontrollers in new designs or provide possibility to add "over-the-air" reprogramming feature to existing applications without necessity of making hardware changes. The presented approach allows a lowering of the power consumption of the nodes and time for node reprogramming comparing with other reported solutions. Summary of our experiences in practical implementation using Texas Instruments (TI) EZ430-RF2500 and CC2430DK development boards and discussion about some key features of suggested "over-the-air" reprogramming method are included.**

I. INTRODUCTION

During recent years Wireless Sensor Network (WSN) technology has developed greatly, allowing to create a wide range of different applications: from small in-flat sensor networks for security or climate control to huge sensor networks distributed over the cities. One common thing for most WSN systems is that sometimes it is required to make changes to sensor nodes initial software. The reasons for changing the software of a node can be different: fixing software errors, changing the communication protocol, modifying the tasks of individual nodes or entire network, or even the movement of the whole network to another place with different conditions [1].

In traditional WSN systems, the only way to change the software of the nodes is to do it manually for every node. So, this operation would require a lot of time and expenses to access every node that should be reprogrammed and is not always practical in real conditions (for example if the system is deployed in hard-reachable or dangerous location). However during recent years different methods for wireless or "over-the-air" programming (OAP) of nodes in WSN were suggested, which allow to change the software of the nodes of the whole network in several minutes without direct accessing the nodes.

The realization of OAP is a hard task in WSN due to many contradictory requirements for such systems. First of all, as the nodes in WSN often have limited amount of energy, it is necessary to minimize the power consumption during reprogramming. This can be achieved, for example, by minimizing the amount of transmitted information and using more reliable transmitting channels or error correcting to avoid data retransmission. The second requirement for such systems is that the new software received by a node should be absolutely reliable and should reach every node in WSN [2]. Besides, it is usually very hard to predict the topology of a WSN and radio network situation for different connections in WSN, but reprogramming should work for all possible cases.

Numerous solutions and protocols for WSN reprogramming realization have been suggested and reported. The main protocols that today represent the state of the art for OAP are Deluge[1], Synapse[2], and MNP[3]. These protocols are now often used and there are regularly appearing publications considering improvement of different aspects of these protocols or with the suggestion of some new protocols.

Although, a significant improvement of WSN reprogramming protocols was reached during recent years, there were not many publications considering possible improvement of WSN nodes hardware architecture. Most of the hardware platforms that are today used with OAP (Mica2, MicaZ, Telos, Tmote Sky, IRIS etc.) have not changed greatly since the first steps for OAP although the electronics had already moved further, allowing to improve the design of the nodes.

In this article, we suggest and evaluate a different hardware architecture of WSN node for OAP that allow to lower the power consumption of the nodes significantly comparing with most often used for OAP WSN hardware platforms. Although our approach is intended for applications where the power consumption is the key-point (like the systems, which are working from the batteries or harvest energy from environment), suggested solution can be used in any system requiring OAP possibility.

II. RELATED WORK

Today the most general WSN nodes usually have architecture similar to one presented in Fig. 1.

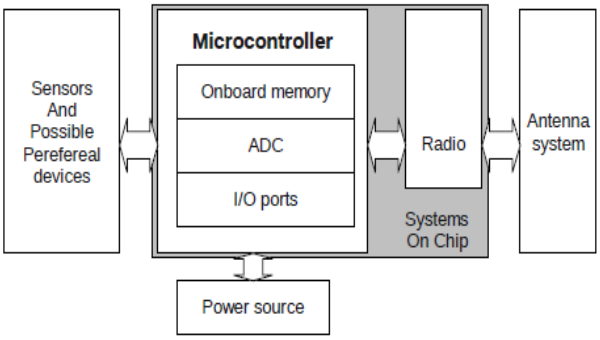

Fig. 1. Typical structure of a WSN node without OAP possibility

As shown in Fig. 1 the WSN node usually consists of a controlling device (typically a low power microcontroller), a radio transceiver chip (microcontroller and radio transceiver can also be implemented in one chip – System-on-Chip(SOC)), power supply (often batteries), some sensors and other peripheral devices depending on application. To save limited power resources, the microcontroller of WSN node usually stays for most of the time in some sort of low power mode, periodically "waking up" to read data from sensors or to send or receive data via radio. WSN nodes are typically both price and power consumption sensitive devices meaning that in-build memory (usually Flash) of microcontroller is used for program storage and external components are not desired.

One of the main difficulties with such WSN node reprogramming implementation is that generic microcontrollers are usually unable to both run the stored program in in-build memory and write to this memory at the same time, which is required during reprogramming. This fact has lead to the situation that up to now two main approaches to the realization of WSN OAP of general microcontrollers has been reported and used. The former suggest usage of a second microcontroller for controlling the radio during the reprogramming of the main microcontroller. The latter uses an external memory chip for the temporary storage of received over the radio program and for rewriting the microcontroller program with this data as soon as the whole program would be received.

The second of these approaches (the structure of such node is presented on Fig. 2) is used in most of the hardware platforms for WSN with OAP possibility such as Mica2, MicaZ, Telos, Tmote Sky etc, which were used for testing most of today OAP implementations [1-7]. Our work was partly inspired by [8], where was presented a table of current consumption during different operations for some of commonly used OAP platforms – see Table I.

From Table I it can be seen that the current consumption for working with external memory of the node is rather high and is comparable to the current required for receiving the program over radio. Besides, if external memory would not be switched off completely after reprogramming, it would add some consumption during the node sleep mode. For comparison: the current consumption in the deepest sleep mode of STM25P80 Flash-memory chip (that is used in Telos node) is at least 20% of overall sleep mode power consumption of Telos node [9].

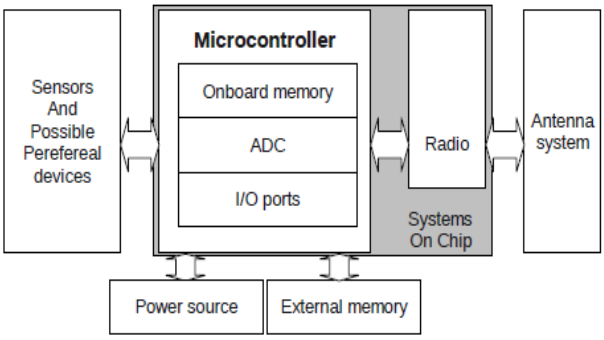

Fig. 2. Typical structure of a WSN node with OAP possibility

TABLE I
CURRENT CONSUMPTION FOR OPERATIONS DURING OAP[8]

Operation	Mote		
	Telos	**Mica2**	**MicaZ**
Node standby (Real-time clock on)	5,1 µA	19 µA	27 µA
Microcontroller (MCU) Idle (Digitally-controlled oscillator on)	54,5 µA	3,2 mA	3,2 mA
MCU Active	1,8 mA	8 mA	8 mA
MCU Active +Radio RX	21,8 mA	15,1 mA	23,3 mA
MCU Active +RadioTX (0 dBm)	19,5 mA	25,4 mA	21 mA
MCU Active +Flash Read	4,1 mA	9,4 mA	9,4 mA
MCU Active +Flash Write	15,1 mA	21,6 mA	21,6 mA

So, it can be seen, that usage of external Flash-memory during reprogramming has serious impact on overall power consumption for node reprogramming and can influence the power consumption during node sleep [10]. Also the addition of an external component to WSN node design would definitely increase the price and size of the node. Besides, until now it was necessary to plan OAP possibility beforehand - already on WSN node hardware developing stage -to have external memory chip installed. Above-mentioned inspired us to propose the new approach for OAP implementation that allows excluding of an external memory chip from the design, which leads to reduce in the price, the size and power consumption of the node.

III. OVER-THE-AIR REPROGRAMMING IMPLEMENTATION SUGGESTION

Let us consider a generic low-power microcontroller. As an example of such microcontroller, we would use MSP430x2xx microcontroller [11, 12], whose structural diagram is presented in Fig 3.

From Fig. 3 one can see that in this microcontroller there are two available memory banks: Flash ROM memory that is usually used as program memory and RAM memory that

stores the data that is used by the program. Nevertheless, this is not the only possible function of RAM memory.

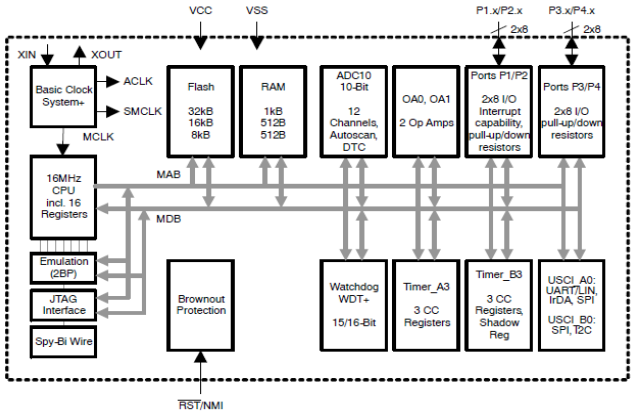

Fig. 3. MSP430 Architecture [11]

One feature that most microcontrollers have and which can be effectively used for implementing OAP, is the usage of temporary memory banks (like RAM) for running the microcontroller program while normal program memory is rewritten [13].

The main advantages of this approach are:

- the suggested approach gives the possibility of executing a program and writing to microcontroller program memory at the same time,
- the suggested approach can be used for a very wide range of microcontrollers, as well to old and to new ones,
- the suggested approach does not require any external components for its realization.

However there are also some features and limitations for this method. There are three main factors that limit the possibility and efficiency of using RAM memory for OAP:

- the size of RAM in microcontroller,
- the complicity of used wireless communication and OAP protocol,
- amount of features that are realized in hardware on microcontroller and radio chip.

The first and the main limiting factor for OAP implementation using the suggested approach is the size of RAM memory, as usually it is much smaller than the size of program Flash-memory. TI EZ430-RF2500 board use MSP430F2274 microcontroller which has 16 kbytes of Flash-memory and only 1 kbyte of RAM memory (more powerful microcontrollers, like TI MSP430F5418 can have up to 16 kbytes of RAM memory). Meanwhile, this memory must contain the program that should allow realizing radio receiving and transmitting, the checking of data correctness (the whole communication and OAP protocols) and writing of received data to location in Flash-memory.

This difficulty can be partly solved by using a part of Flash-memory (which would not be rewritten during reprogramming) for the storing OAP protocol program. In this case, at the beginning of reprogramming, to special non-rewritable location in Flash-memory would be copied the program, realizing OAP protocol. This program would be used for initiating OAP, receiving radio packets, checking received packets and sending the acknowledgements (ACK). As soon as OAP program would receive and check a packet, it would call a special small program that is stored in RAM memory, which will rewrite the necessary Flash-memory block with received data. As soon, as writing would be finished, the control will be given back to the program in Flash-memory. Obviously, the described method would not allow making a complete reprogramming of the whole Flash-memory, as it is necessary to keep untouched the temporary block from which the OAP would be running. However, the usage of Flash-memory for OAP protocol allows removing the limitations on the size and complicity of used OAP protocol which are defined by the size of RAM memory when the whole program is stored in RAM. Also this approach allows having an effective "back up" program in the case, if something would go wrong during reprogramming.

Two above described approaches are rather general and can be implemented for most microcontrollers, which are used in WSN applications now. The third option, that can be used for some chips (like CC2430) requires the presence of a direct-memory access (DMA) controller, which is a part of many of contemporary microcontrollers and SOC. Usage of DMA channel for coping data from RAM to Flash allows to realize OAP without using RAM memory for storing the program at all. However at the same time, it is impossible to continue to execute the node program while Flash-memory is cleared or written in this case, unlike when rewriting Flash-memory using the program in RAM. Besides, this method is not as universal as two previous and can be implemented only for microcontrollers with the in-builded DMA controller (for example MSP430F2274 does not have DMA controller).

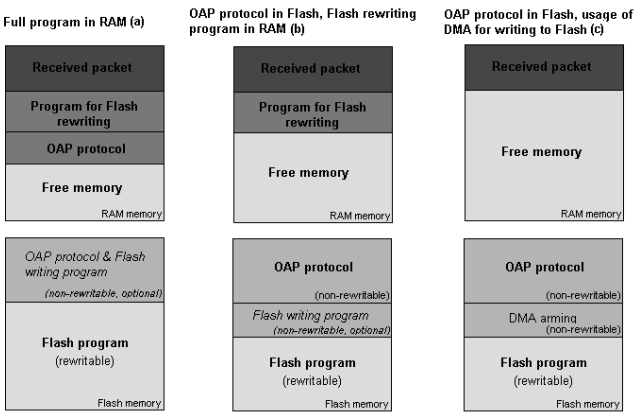

Fig. 4. Microcontroller memory allocation for suggested approaches realisation: for program fully stored in RAM(a), program in non-reprogrammable Flash part with RAM program used during Flash writing(b) and program in non-reprogrammable Flash part and using DMA for Flash rewriting(c)

The second factor that should be considered when developing a program using described approaches is the unavailability of interrupts when Flash-memory is rewritten. The reason for it is that interrupt vectors for microcontrollers

should be on specific positions inside Flash-memory. So, if the application requires continuous monitoring of environment (for example in hazard detection application), this should be done by the program running in RAM.

The third factor is defined by used architecture and should be considered when OAP protocol for such system is developed. Old systems with external memory chips were able firstly to store the whole new program and then to change the whole program at once. For suggested architecture, as there is no place for temporary storing the program any more, program pages should be written to microcontroller Flash as soon as they are received and checked. This fact makes it harder to keep a back-up copy of the whole node program (although it can be possible if the size of on-chip Flash-memory is at least 2 times bigger, than the size of program) and makes it hard to stop the reprogramming process once it had started.

Finally, RAM memory is power dependent – with power reset the data in RAM memory will be lost. Thereby designs with a program in RAM are very sensitive to power downs. So, if power downs are possible (like, for example, in systems with energy harvesting), one should implement restoring program that would be in non-rewritable part of Flash and that would be launched after power supply would be restored.

IV. Hardware Evaluation of Suggested Approaches

To evaluate described above approaches we used two different platforms. The first was TI EZ430-RF2500 development boards, which encapsulate MSP430F2274 microcontroller and CC2500 radio chip[14]. The second - CC2430DK development boards with based on 8051 microcontroller CC2430 SOC [15]. The programs were developed using IAR Embedded Workbench without any operational system (OS) (difference with regard to most of the contemporary OAP protocols that usually use TinyOS or some other OS[1-2]). There were two main reasons for not using OS. First one is limited amount of microcontroller resources, especially if OAP protocol would be stored fully in RAM memory. Secondly - although the usage of OS makes the programming of the nodes simpler, OS requires additional computational power and would increase the overall power consumption of a node.

The design of EZ430-RF2500 board appeared to suit rather good for implementing suggested OAP approaches – microcontroller and radio chip already have some realized in hardware features that allowed lowering the size of program. For example, the radio chip on the board is connected to the microcontroller via Serial Peripheral Interface (SPI) interface, the drivers for which in microcontroller are realized in hardware. Also, some network level features like address or error checking are already included and implemented in hardware of the radio chip [16]. CC2430DK boards are also quite suitable for OAP realization. As CC2430 is SOC, it is not necessary to transfer data from radio to microcontroller, because microcontroller can access radio registers directly. Also one of the very useful features of CC2430 that was used during OAP was DMA. Usage of DMA controller for transferring data between RAM and Flash memories allowed to launch memory writing directly from Flash program, without using RAM.

Our main task was to check if it is possible to use suggested approaches for simplifying hardware architecture and to evaluate power consumption compared with existing solutions with "classical" architecture. Therefore in our tests we used a very simple OAP protocol that had only the most important features.

The principal diagram of the developed test application program is presented on Fig. 5. Developed program implements simple control over radio for:

- receiving reprogram data packets,
- checking the correctness of a received packet (using implemented in radio hardware features like cyclic redundancy check, CRC),
- sending negative acknowledgement to the base station to show that the packet was received with errors (similar to Deluge protocol),
- writing data to Flash-memory of microcontroller if the received packet was without errors.

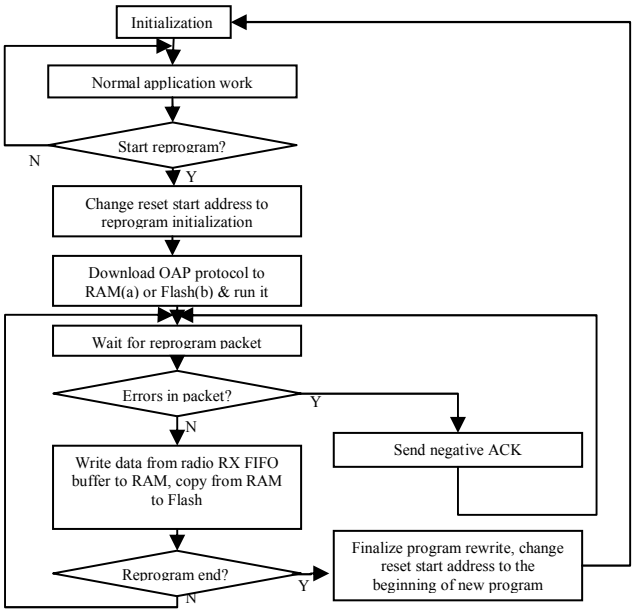

Fig. 5. Algorithm of test program

Such simple protocol with the usage of hardware implemented features whenever possible, allowed us to make the size of developed program very low. So, for EZ430-RF2500 development boards when OAP protocol was working from RAM (like in Fig 4. a) the size of program is as low as 208 bytes for Initialization Subprogram, 28 bytes for coping the program to RAM and 476 bytes (around of 46% of available RAM memory for MSP430F2274 microcontroller, including the buffer of 64 bytes for storing the received reprogram packet) When OAP protocol was working from Flash-memory on EZ430-RF2500 (like in Fig 4. b), the size of used memory was 352 bytes of Flash for OAP protocol (including the copy of the program which is running in RAM

for back-up realization) and 180 bytes of RAM (including the buffer of 64 bytes for storing the received reprogram packet). The size of program, with DMA (like in Fig 4. c) for CC2430 based boards, was 3832 bytes (around 6% of the available). As shown in all cases the amount of used memory is quite small, which allows to make the significant improvements of OAP protocol in future or to use more complicated OAP protocols using the same approach.

To check the efficiency of the suggested OAP approach compared with already existing, we measured the energy consumption for developed programs. The curves of current consumption during reprogramming for RF2500 boards are presented on Fig. 6 and for CC2430 – on Fig.7. The measured average values for different operations during reprogramming for both hardware platforms are presented in Table II. In Table II are also presented the values for operations with external Flash-memories (STM25P80, that is used in Telos nodes and Atmel AT45DB041D in Mica nodes) which would be required for OAP realization using "old" architectures.

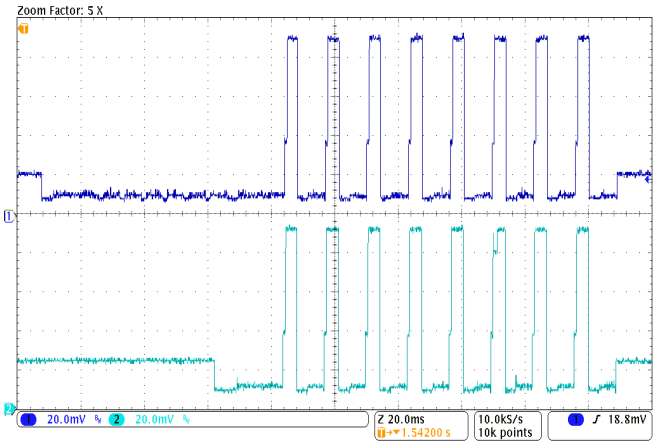

Fig. 6. Current consumption for reprogramming one Flash page (512 bytes) for RF2500 boards. (Measurements were made for voltage drop on 4.7 Ohm resistor using TEK MSO4054 oscilloscope, channel 1 – OAP protocol in Flash-memory, channel 2 – OAP protocol in RAM)

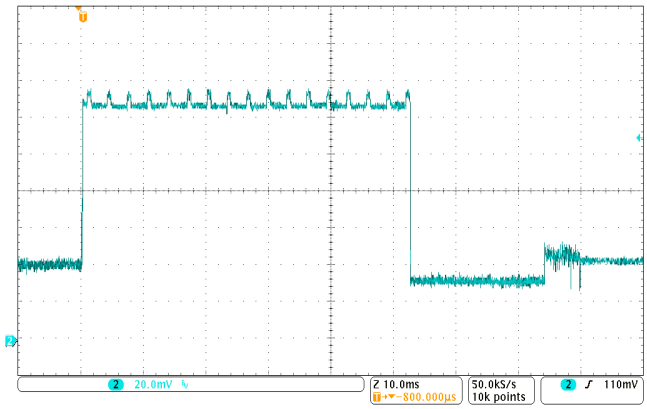

Fig. 7. Current consumption for reprogramming one Flash page (1024 bytes) for CC2430 boards. (Measurements were made for voltage drop on 4.7 Ohm resistor using TEK MSO4054 oscilloscope)

TABLE II
POWER CONSUMPTION DURING OAP

Operation	Consumption		
	Average current, mA	*Time, ms*	*Energy, µJ*
Preparation for reprogramming:			
Copy OAP protocol to RAM [a]	1,7	3,54	15,1
Copy OAP protocol and RAM program to "temporary" Flash, download program to RAM [b]	2,04	62,82	320,1
Erase microcontroller Flash sector (512 bytes for RF2500/1024 bytes for CC2430)	2,15 6,7[d]	14,5 21,6[d]	77,9 376,4[d]
Reprogramming (64 bytes):			
Receive 64 bytes over radio (incl. switching) + OAP protocol in RAM [a]	16,63	2,61	108,5
Receive 64 bytes over radio (incl. switching) + OAP protocol in Flash [b]	16,70 28,22[d,e]	2,61 3,22[d,e]	109,0 236,3[d,e]
OAP protocol in RAM+ Idle Radio [a]	1,70	3,4	14,5
OAP protocol in Flash+ Idle Radio [b]	1,76 4,3[d]	3,4 -	15,0 -
Write page (64 bytes) to microcontroller Flash, Program working in RAM [a,b]	2,13	5,32	28,3
Send NACK packet over radio (including switching Idle->TX) + OAP protocol in RAM [a]	7,89	2,52	49,7
Send NACK packet over radio (including switching Idle->TX) + OAP protocol in Flash [b]	7,92 23,19[d]	2,52 0,92[d]	49,9 55,5[d]
Overall(for receiving 64 bytes, packet is ok) OAP protocol in RAM [a]	5,34	11,33	151,3
Overall(for receiving 64 bytes, packet is ok)OAP protocol in Flash [b]	5,38 28,22[d,e]	11,33 3,22[d,e]	152,3 236,3[d,e]
Reprogramming of a Flash page (512 bytes for RF2500 and 1024 bytes for CC2430):			
OAP protocol in RAM (only reprogramming/including preparation) [a]	5,34 /4,8	90,64 /108,68	1210,3 /1303,3
OAP protocol in Flash (only reprogramming/including preparation) [b]	5,38 /3,85 21,0[d] /21,0	90,64 /167,96 78,76[d] /78,76	1218,1 /1616,1 4298,7[d] /4298,7
External memory (512 bytes): [9,10,16,17]			
Write to Flash (STM25P80) (for operations with pages)[c]	16,02 17,5	7,16 7,16	298 326
Read from Flash (STM25P80) (for operations with pages) [c]	5,7 8,3	4,16 4,16	62 90
Sector erase (STM25P80)(64 kbytes)	15	1000	39000
Write to Flash (AT45DB041D)(for operations with pages) [c]	12,75 14,2	8,22 8,22	272 304
Read from Flash (AT45DB041D) (for operations with pages) [c]	8,7 11,3	4,22 4,22	96 124
Sector erase (AT45DB041D) (64 kbytes)	12	1600	49920
Low power modes:			
MSP430 Microcontroller + CC2500	0,0012		
CC2430 SOC	0,0005		
STM25P80 (deep sleep)	0,001		
AT45DB041D (deep sleep)	0,015		

a. For implementation with OAP protocol in RAM (Fig.4, a)

b. For implementation with OAP protocol in Flash-memory (Fig.4, b,c)

c. Data for External Flash-memory is given with 1MHz clock frequency (which was used for microcontroller) and includes microcontroller power consumption (microcontroller sleep mode used whenever possible)

d. The values in "shaded" cells are for CC2430 development boards, in white – for EZ430-RF2500

e. The values include the processing of the packet and writing it to RAM memory of CC2430

Here it is necessary to explain the choice of some parameters, which affected the work of our application and received results. Both boards, which we used for practical implementation, have radios working in 2.4 GHz ISM band. However, RF2500 uses a proprietary radio protocol stack (in our case – MSK modulation and 250 kbit/s data rate) while CC2430 uses physical layer of 802.15.4 standard [18]. For both cases, each sent packet contained 64 bytes of reprogramming data. The reasons for using this packet size are that this amount of data allows to leave enough space for all sorts of headers for 802.15.4 standard and additional information for packet correctness checking. Also, 64 bytes is the maximum size of data page that can be written using a single command in MSP430 microcontroller Flash. The size of Flash page during erasing for MSP430 is 512 bytes and 1024 bytes for CC2430. Thus we developed the test applications that would rewrite a single page in Flash-memory. Thereby, during testing were sent 8 packets (512 bytes of payload) for RF2500 boards and 16 packets for CC2430 boards (1024 bytes of payload). (See Fig.6 and Fig.7). Presented in Table II measurement results were used for calculating overall power consumption for single page reprogramming using suggested approaches and for calculating the additional power consumption of an external Flash, required if the traditional approach would be used.

V. Comparison of Suggested and Classical Approaches Efficiency

As it can be seen from presented in Table II results, in the case without necessity to retransmit the packets, the power consumption for reprogramming 512 bytes for RF2500 boards would require 1210 μJ of energy, at the same time, if reprogramming would be done using external memory chip it would take additional 360 μJ with STM25P80 chip (suggesting that external memory has been already erased, if considering also the Flash erase with the sector erase – 664 μJ) and 366 μJ with Atmel AT45DB041D chip (756 μJ considering the Flash erasing). So, the profit on energy consumption for suggested OAP hardware architecture would be 23% (with Flash erase – 38.5%) comparing with traditional solution with Atmel AT45DB041D chip and 23% (with Flash erase – 35%) with STM25P80 chip and the profit in required for reprogramming time would be around 9% (17.5% considering Flash erase) for STM25P80 and 12%(21.6% with Flash erase) for AT45DB041D chip. (The values are given only for reprogramming, not considering preparation procedures). For CC2430 boards, the page reprogramming with 1024 bytes requires 4300 μJ of energy. The additional power consumption for using external memory chips would be 651 μJ for STM25P80 chip (1260 μJ with Flash erasing) and 608 μJ for Atmel AT45DB041D chip (1388 μJ with Flash erasing). In this case, the profit in power consumption due to excluding external memory would be 12.4% (with Flash erase – 24.4%) for Atmel AT45DB041D chip and 13.1% (with Flash erase – 22.6%) for STM25P80. The profit in time for reprogramming for the CC2430 based system is around 22% (32% with Flash erase) for STM25P80 and 24% (39% considering Flash erase) for AT45DB041D chip.

Besides, as it can be seen at Table II, great profit for power consumption in sleep mode is achieved, that is often even more important, as a WSN node usually spends most of the time in this mode. For RF2500 boards this profit would reach 45% comparing with an application with STM25P80 memory chip and 94% comparing to an application with AT45DB041D. For CC2430 – 67% and 97% respectively. The other profit of excluding external memory chip from WSN node is the supply voltage, which is needed for the system. Considered external memory chips require to have at least 2.7 V supply voltage during writing, while writing to internal microcontroller memory can be done with supply voltages above 2.2 V.

Another result which can be seen at Table II is that the time and power consumption for reprogramming in RF2500 for tested approaches (when OAP protocol was in RAM and in Flash-memory) are very close, but the program in RAM consumes slightly lower power (around 0.6% lower) than the one in Flash-memory. This result is caused by lower power consumption for the microcontroller program running from RAM comparing with the program, running from Flash-memory. By our tests, the difference in power consumption for single microcontroller with the program running from RAM and from Flash is around 10-15% for MSP430 and around 15-20% for CC2430. This fact can be used for further improvement of power consumption in systems with harsh energy conditions and can give significant improvement if the system requires many similar operations during data processing.

Also, it can be seen that the preparation for running OAP program from Flash for RF2500 boards requires much more time due to the necessity of coping OAP protocol to Flash-memory. (This can be fixed by keeping Flash program in non-rewritable Flash, but in that case it would be impossible to change OAP protocol wirelessly at all). Usage of DMA for transferring data between RAM and Flash-memory does not require any special preparation procedures at all and can be initialized directly from OAP Flash-memory.

VI. Discussion and Conclusions

"Over-the-air" reprogramming (OAP) feature for Wireless Sensor Network (WSN) is now becoming one of the substantial things in modern designs. This article describes a potential, new, and energy efficient hardware approach for implementing OAP in WSN. Suggested in this article new method is based on the usage of microcontroller RAM for storing the node program while reprogramming is performed.

Two possible variants of this method were presented and evaluated: the first method suggests placing the whole OAP protocol in RAM memory, the second uses stored in non-rewritable during reprogramming Flash-memory OAP protocol and small program in RAM for coping received data to Flash-memory. Usage of second method allows overstepping the limitation on the size and complicity of OAP protocol caused by the size of RAM memory for the first approach and allows realizing an effective back-up at the same time, but this method

requires significantly longer and power consumptive "preparation" cycle due to the necessity of coping Flash-memory parts. We have also evaluated the modification of second approach based on using DMA for transferring received data from RAM to Flash. Although this method requires microcontroller containing the in-builded DMA controller, it does not require any program running from RAM to copy data to Flash, which simplifies the development of the program and preparation for reprogramming. The main disadvantage of this method is that using it, it is impossible to track and reply to any events, while Flash-memory is being modified.

For evaluating suggested approaches, were developed applications using two WSN platforms, based on MSP430 and 8051 microcontrollers and two types of radios: FSK with proprietary protocol and 802.15.4 standard. The fundamental tests of possibility of program memory rewriting with a program running from other memory were also done for Atmel ATmega128 microcontroller (there instead of RAM memory was used a Boot section block in Flash-memory)[19]. Thus we are convinced that suggested approaches are applicable to most popular microcontrollers which are used today for WSN platforms. Besides, suggested solution allows adding OAP feature to some already existing applications without necessity of changing the hardware design.

The main advantage of our approach over other reported OAP implementations is that it does not require any external components, like second microcontroller or external memory chip. This is very critical due to the typical WSN node requirements of low power consumption and cost efficiency. According to our measurements, the possible improvement for reprogramming power consumption and reprogramming time due to excluding external Flash-memory chip from the design are 10-40% for power consumption and up to 40% for time compared with the designs containing external memory chip, depending on microcontroller and memory chip characteristics. Besides, removing external memory from the design allows also lowering the power consumption during the sleep mode (for reviewed designs the profit was 45-97%, depending on memory chip characteristics) as well as lowering the size and the price of a node.

Although suggested approaches were tested only for a simple proprietary OAP protocol, we are convinced that the same idea can be used with the wide range of already developed OAP protocols such as Deluge, MNP, Synapse, etc. Such solution would allow designers to combine the features of these protocols (like reliability, scalability, etc.) with the low power consumption of the suggested hardware approach. This, together with the practical implementation of WSN OAP using suggested approaches for other platforms would be our future tasks.

ACKNOWLEDGMENT

The authors wish to thank all companies and other parties who have either participated or financially supported this study. This work has been supported by European Regional Development Fund, Council of Oulu Region, Finnish Funding Agency for Technology and Innovation (Tekes), Ylivieska Subregion, and companies.

The authors also wish to thank Professor Markku Juntti from University of Oulu and anonymous reviewers for valuable contributions and comments on earlier versions of this article.

REFERENCES

[1] A. Hagedorn, D. Starobinski, and A. Trachtenberg, "Rateless Deluge: Over-the-Air Programming of Wireless Sensor Networks using Random Linear Codes", in *Proc. IPSN 2008*, pp. 457 – 466, 22-24 Apr. 2008.

[2] M. Rossi, G. Zanca, L. Stabellini, R. Crepaldi, A.F. Harris and M. Zorzi, "SYNAPSE: A Network Reprogramming Protocol for Wireless Sensor Networks using Fountain Codes", in *Proc. SECON 2008*, pp. 188 – 196, 16-20 June 2008.

[3] S. Kulkarni and L. Wang, "Energy-efficient multi-hop reprogramming for sensor networks", *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, issue 2, pp. 1-40, Mar. 2009.

[4] (2010) Texas Instrumants: Over the Air Download Evaluation Module Kit. [Online]. Available: http://focus.ti.com/docs/toolsw/folders/print/oad-emk.html

[5] C. Chen, Y.Huang and C.Zhang, "Toward a Real and Remote Wireless Sensor Network Testbed", *WASA 2008*, LNCS 5258, pp.385-396, 2008.

[6] S. Varma, U.S. Tiwari and R Konakalla, "Remote reprogramming mechanism for WSN", *ICIAS 2007*, pp. 939 – 944 , 25-28 Nov. 2007.

[7] A. Dunkels, N. Finne, J. Eriksso and T. Voigt, "Run-Time Dynamic Linking for Reprogramming Wireless Sensor Networks " in *Proc. of the 4th international conference on Embedded Networked Sensor Systems*, pp.15 – 29, 31 Oct. – 3 Nov. 2006.

[8] J. Polastre, R. Szewczyk and D. Culler, "Telos: enabling ultra-low power wireless research", in *Proc. IPSN 2005*, pp. 364- 369, 25-27 Apr. 2005.

[9] "M25P80 datsheet", ST Microelectronics, Geneva, Switzerland

[10] G. Mathur, P. Desnoyers, D. Ganesan and P. Shenoy, "Ultra-low power data storage for sensor networks", in *Proc. IPSN 2006*, pp. 374-381, 19-21 Apr. 2006.

[11] "MSP430x2xx Family Users Guide (Rev. E) (SLAU144E)", Texas Instruments, Dallas, Texas, USA.

[12] S. Corroy, J. Beiten, J. Ansari, H. Baldus, and P. Mahonen, "Energy Efficient Selection of Computing Elements in Wireless Sensor Networks", in *Proc. SENSORCOMM 2008*, pp. 312 – 318, 25-31 Aug. 2008.

[13] "MSP430 Flash Self-Programming Technique (SLAA103)", Texas Instruments, Dallas, Texas, USA.

[14] "EZ430-RF2500 Development Tool User Guide (Rev. E) (SLAU227E)", Texas Instruments, Dallas, Texas, USA.

[15] "CC2430 Data Sheet: A True System-on-Chip solution for 2.4 GHz IEEE 802.15.4 / ZigBee® (rev. 2.1)) (SWRS036F)", Texas Instruments, Dallas, Texas, USA.

[16] "Low-Cost Low-Power 2.4 GHz RF Transceiver (Rev. C) (SWRS040C)", Texas Instruments, Dallas, Texas, USA.

[17] "AT45DB041D datasheet", Atmel Corporation, San Jose, California, USA.

[18] IEEE 802.15.4 Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), IEEE Std. 802.15.4, 2006.

[19] "Atmel's Self-Programming Flash Microcontrollers", Atmel Corporation, San Jose, California, USA.