Position papers of the 2014 Federated Conference on Computer Science and Information Systems pp. 191–196
DOI: 10.15439/2014F155
ACSIS, Vol. 3

Toward implementing an efficient gateway for wireless sensor networks

Gundars Miezitis,
Riga Technical University
Faculty of Computer Science and Information Technology
Meza str.1/321 LV-1007 Riga, Latvia
E-mail: gundars.miezitis@rtu.lv

Romans Taranovs,
Riga Technical University
Faculty of Computer Science and Information Technology
Meza str. 1/3-322 LV-1007 Riga, Latvia
Email: romans.taranovs@rtu.lv

***Abstract*—Timely and energy efficient data delivery is important in wireless sensor network applications. To reduce the probability of wireless sensor network disconnection from user a lower energy consumption gateway must be used, but data delivery speed should be maintained as high as possible. So our purpose is to compare and analyze common approach of building gateway for wireless sensor network applications, where high performance ARM boards are used as gateways, with less common approach, where simple eight bit microcontroller boards can work as gateway. We develop two similar test-beds using two available boards – DiGi Wi-9C (high-end) and ATXmega Xplained-A1 (low-end). We test both boards using the same data processing algorithm and by measuring delivery speed and energy consumption we make conclusions. Contrary to our expectations simple eight bit microcontroller showed even better results than we had expected. While this board consumed less energy it guaranteed faster and more stable (little or no delivery speed deviations) data delivery. Thus we concluded that using simpler hardware can not only reduce energy consumption, but ensure high data delivery speed as well.**

***Keywords*—sensor network, gateway, data transmission speed, energy efficiency**

I. Introduction

WIRELESS sensor networks (WSN) enable subtle monitoring of the environment, buildings and human activity. Built from small devices about the size of a matchbox, called sensor nodes, that compute, relay data to each other and sense phenomena. This allows a user to monitor objects of interests for long periods of time and in greater detail. Developing energy efficient protocols and algorithms for WSNs has always been an important research question, but it is essential to improve the WSNs connectivity to other networks for increased usability and WSN system value. By interconnecting WSN with a global network, like the Internet, GSM, accessing WSN data can be transmitted from distance – another building, country or even continent. Thus bringing closer to realization of Internet of Things (IoT) [8].

To enable network interconnection the transition from one network to the other is required. One of possible options in WSNs interconnection is by using gateway (GW) – which basically translates one network data stream to other and vice versa and additionally can perform data aggregation. In our study-case it is between TCP/IP and WSN. GW usage relieves work load from sensor nodes and enables WSN to use specialized protocols, as well WSNs are more scalable and elastic to changes [1]. To our knowledge very few different gateway approaches have been proposed [2,5,7]. They include a) designed application layer GW for interconnecting WSN and TCP/IP networks using PXA270 board [2], b) an ARM based GW for interconnecting WSN and TCP/IP [5] and c) Samsung S3C2440 400MHz CPU [7]; all of which have been based on high-end devices. No justification or guidelines have been provided for the choice of hardware for GW.

The lack of a justification may be due to different application requirements or researchers assumption that GW has unlimited energy resources. But there are applications, mainly outside of cities, where gateways can't be plugged in and batteries must be used for GWs as well. One such application we can mention is equipping tractor with sensors to achieve autonomous work execution and provisioning users with most relevant and fresh data. Thus we are interested in – how the hardware choice for GW impacts performance and how it influences WSN data transmission when communicating with user, and network lifetime as well. Furthermore we wish to provide some easy to apply guidelines that could be used to make the right choice in GW hardware.

Most common approach for building WSN GW is to use ARM processor based boards (or other similar high-end device) running OS, like Linux; this was done in [2,5,7]. From our experience we know that this approach is more profitable – because use of OS can reduce application development expenses and complexity; furthermore, if constant energy source is available high-end board is a logical choice. But if we consider WSN system to be energy constrained, that includes GW as well, choosing or constructing appropriate board must be done carefully. As far as we know we are first to compare hardware that is used in building energy constrained GW for WSN. This may be due to fact that researchers want to check their developed approach overlooking energy efficiency in GW.

As we mentioned common approach used for GWs, is based on high performance ARM (32-bit) processors. But there

exists a high-performance low-energy consumption 8-bit microcontroller based boards that could be used for GWs as well. The comparison of two selected board's performance under load conditions, from each end, is main attraction of this research and conclusions are based on obtained results. By comparing both boards we hope to obtain results that would clarify choice of hardware and if it can significantly affect system operation altogether and relive choice among available boards leading to more suited GW for.

The rest of the paper is organized as follows. In II we describe related work. In III we describe constructed test beds and software for GW testing and methods used for evaluation. In IV we show and analyze obtained results. In V we have implemented similar test beds for more detailed comparison. And finally in VI we conclude our research.

II. Related Work

GWs provide simple and easy to implement interconnection among two different networks by using intermediate node between these two networks. In [2] is designed application layer GW for interconnecting WSN and TCP/IP networks using PXA270 board. In [4] is proposed network interconnection address mapping mechanism, simple structure of GW node and mechanism for multiple GW selection. Paper focuses on theoretical network interconnection and GW structure. In [5] is proposed yet another ARM based GW for interconnecting WSN and TCP/IP. In [6] are discussed GW mobility aspects when interconnected with GWs. And in [7] Samsung S3C2440 400MHz CPU is proposed for WSN GW.

Alternatives include using more complicated, but more transparent method – TCP/IP protocol over sensor nodes as presented in [1,4]. In [1] is presented TCP/IP on sensor nodes a.k.a., overlay concept, for WSNs. In [3] SunSPOTS are described that are powerful sensor nodes (based on ARM processor, running at 200 MHz) and runs TCP/IP stack. This approach uses sensor nodes equipped with a TCP/IP protocol stack. By implementing TCP/IP stack on sensor nodes, border between WSN and TCP/IP network becomes blurry and often is referred as "Internet of Things" [8]. Drawbacks include increased energy consumption and communication overheads – because of TCP/IP packet size, due to noisy environment links between nodes are unreliable, but TCP/IP has been created as reliable protocol stack. Among advantages are routing and medium access – they are fully developed and should be easily adapted.

III. Description of test beds

For our research purposes two slightly different test beds were proposed. Common for both test beds are – sensor nodes, gateway (interconnected controller and sink node) and main computer, located in local network. Due to low energy consumption – ultra-low-power MCU, low-power radio module and protocol stack – SimpliciTI – eZ430-RF2500 boards were chosen as sensor nodes [9]. Connection between sensor nodes and gateway are wireless, but connection between gateway and main computer can be both – wireless or wired, as long as it ensures TCP/IP stack. Due to implementation simplicity wired – TCP/IP – connection setting was used.

To enable gateway connection with sensor nodes, controller board must be connected with one sensor node, called sink node, trough available hardware communication interface – this is widely used approach in GW interconnection. For us two possibilities exist – SPI (Serial Peripheral Interface) and USART (Universal Synchronous Asynchronous Receiver Transmitter). SPI is more suited for applications that require faster data delivery, while USART is more common communication interface and will be frequently available, but with more communication overhead. SPI speed is directly affected by boards source clock and generally can be derived as – maximum transfer speed is half of used clock speed (this is true for selected boards, but not for all microcontrollers/processors in general). Due to fact that sensor node maximum clock frequency is 16 MHz, maximum SPI transfer frequency is chosen as 8 MHz. But using higher sensor node clock frequency will lead to faster energy source depletion – developer must choose from these tradeoffs.

Based on common test bed assumptions DiGi Wi-9C board test bed is depicted in Fig. 1. and Xplained board test bed is depicted in Fig. 2. The difference can be seen in Fig. 2. – new board had to be included – ENC28J60-H – because Xplained board doesn't have built-in Ethernet support.

A. Design challenges

After choosing all hardware to test implementation, a few design challenges had to be overcome, as described below.

First was connecting sink node to gateway controller. No free SPI interface was available, because in Texas Instruments board drivers interface was used for other purposes. One possibility is to implement software SPI to connect both boards. Drawback of this is that transfer of data is slower than it is possible with hardware SPI. Second was to introduce SPI imitation mode, where incoming data were modulated from board that imitates sensor node functionality with maximum transfer rates.

The second challenge was interconnection of XMEGA XPLAINED-A1 and Ethernet controller ENC28J60-H. No TCP/IP stack was implemented on ATXmega's, but fortunately for ATmegas there were examples - uIP. So we had to port existing TCP/IP example to work with ATXmega. Although this wasn't done perfectly basic communication between PC and XMEGA XPLAINED-A1 was possible and worked without bugs.

Third was related to SPI driver on DiGI board. Available SPI driver for Linux operated only in master mode so both GW controllers were implemented in master mode and sink node as slaves. This means that sink node has to buffer data that are to be forwarded to GW controller so that they are always ready for sending after master node request.

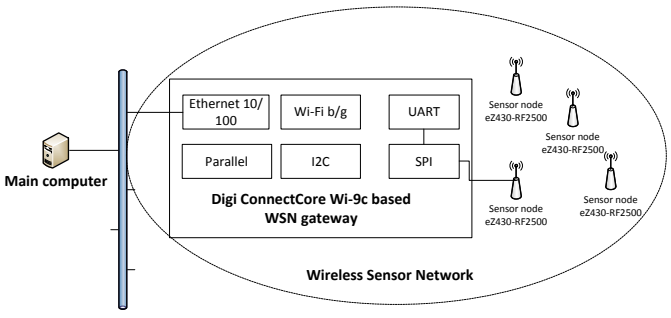


Fig. 1. ConnectCore Wi-9C test bed.

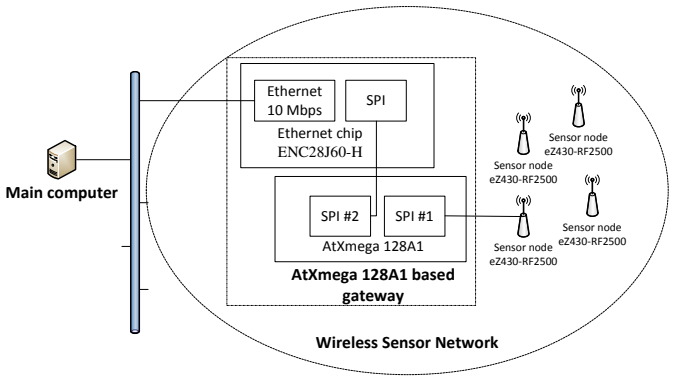


Fig. 2. ATxmega128A1 based gateway integration into WSN an interconnection with main computer.

B. Gateway operation algorithm for testing purposes

Based on proposed test beds, GW operation algorithm was developed that could ensure valid and reliable results. This algorithm is depicted in Fig. 3.

Algorithm executes after following steps:

1) Setting up board – variables, hardware communication interfaces and Ethernet interface.
2) Testing cycle can begin. It is repeated 100 times to ensure large sample count.
   a) Each test cycle consists of reading data from sink node using SPI or USART (depend on executed test). From each sink node 20 bytes are read. Assuming that each sensor node has sent value of 12 bits long (sensor nodes Analog to Digital converter resolution), 10 sensor nodes have sent measured value to sink node.
   b) After reading sensor node data it is inserted in predefined UDP (User Datagram Protocol) packets payload;
   c) And data are sent to user via Ethernet connection.

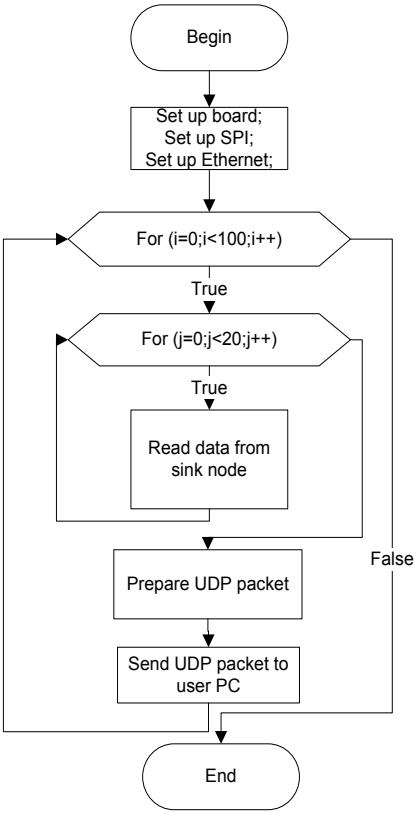


Fig. 3. Gateway test program's flowchart.

Developed algorithm had to be implemented on both boards. Linux API (Application Programming Interface) functions and C/C++ were used due for DigGi Wi-9C while pure C was used for ATXMeaga board. We came to conclusions that using API makes programming less complicated and application development faster.

C. Tools used for data delivery speed measurement

Network protocol analyzer - Wireshark - was used as a tool to measure data transfer speed on user side. By using this application we can see what data we have received trough Ethernet interface and at what speed. It makes data checking and result processing easier.

But using Wireshark can only show total data transfer speed on user PC. If data transfer speed between sink node and gateway controller must be measured reliably, logic analyzer, that is connected to communication wires and captures transferred data, must be used. Here, data transfer time and amount can be measured, using logic analyzers software, to calculate data transfer rate. We used Intronix LA1034 Logicport logic analyzer.

IV. Results and Evaluation

In this section we will discuss main results obtained after testing GW in proposed test beds. In all following figures transfer speed is depicted in **kBps** and time in **ms**.

A. Full communication link speed test

This test evaluates full communication path – from sink node to user PC, and speed performance of both selected gateway boards. Obtained results: Figure 4 a) shows, that by using ATXmega128A1 based GW, data transfer is almost eight times faster than DiGi board in 8MHz imitation mode and more than two times faster in software SPI mode. Even ATXmega128A1 software SPI mode is working faster than DiGi hardware SPI, which implies that ATXmega board can perform data forwarding with less overhead. Furthermore, it was measured that ATXmega128A1 gateway uses around 300 mA while DiGi gateway uses around 800 mA while operating as gateway.

Obtained results were different than what we had anticipated. We had expected that with DiGi board data transfer from sink node to user PC would be faster, but as we can see in Figure 4 a) this assumption was incorrect. To make sure that results are correct, test was performed for four more times and average values were presented. During these tests we noticed that DiGi board values deviates more than ATXmega. But in discussion about deviation we go into more details later. We can see that by choosing simpler hardware board data can be transferred even faster and with less energy consumed.

There are few drawbacks in our tests. First, we are aware that to receive more general results, more than two boards should be compared, but due to resource limitation it is not possible. When paper was prepared Atmel had released even more energy efficient and faster SAM4L boards, which could perform even better in these tests. Second, Linux kernel that was used wasn't fully real time kernel, unfortunately present Linux kernel did not support real time patch. Thus further researches should be done to confirm our results. Third, sensor nodes with free hardware SPI should be used to avoid imitating this connection or implementing software SPI.

As we mentioned earlier common approach is to choose powerful ARM based boards, but when implementing real application gateways these results should be taken into

account, because this could ensure longer network operation and thus smaller network maintenance costs.

B. Sink node to gateway connection

This connection was tested by sending large data amount (4000 bytes) from sink node to gateway controller and average transfer speed calculated and results were depicted in Figure 4 b). DiGi board in hardware/imitation SPI mode present small connection speed improvements compared to software SPI mode – only about two times. While ATXmeag128A1 gateway in software SPI mode already was three times faster than DiGi board and in hardware/imitation SPI mode more than twenty five times faster than software SPI mode.

This might be a little unexpected, but when we observed logic analyzers measured data, the gap in average transfer speeds was clear. ATXmega board had no delays between transmitted sink node bytes while DiGi board had random (at least we didn't notice any regularity) delays between byte transmissions. We want to remind that gateway controller acts as master due to DiGi limitations – it does not support SPI master mode, while ATXmeg board does not have this limitation.

C. Gateway to user PC connection

Connection was tested by sending large amount of UDP packets (100 packets) from gateway controller to user PC and network analyzer Wireshark was used for measuring transfer speed and results are depicted in Figure 4 c). From results it is seen that ATXmega board can transfer data two times faster than DiGi board. When performing this test we noticed that DiGi gateway has great variation in transfer speed, even as much as two times which is depicted in Figure 5 c). From both previous tests we can see that DiGi has more deviations which reduce total delivery speed.

D. Communication speed tests for gateway

These results present same tendency what was seen in full communication cycle, i.e. ATXmega based gateway transfer data faster than DiGi based gateway. This in the end leads to faster total speed.

Due to differences in presented results, we came to conclusion that total transfer time is formed out of three different components, of witch second is indirectly observable:

a) Time to transfer data from sink node to gateway – data transfer using SPI interface;
b) Time to switch between communication streams in gateway – copying data from SPI buffer to Ethernet buffer;
c) Time to transfer data from gateway to user PC – data transfer using Ethernet UDP packets;

Thus introducing additional time that can influence total transfer speed.

E. Switching between communication streams

Based on obtained connection speeds, switching between communication streams was calculated and depicted in Figure 4 d). It was performed equally for all tests, which shows that DiGi board is yet again deviating while aggregating data.

With this our test was concluded and interesting and new results were obtained. Due to limited resources we have, we invite other researchers to perform their own tests and compare results.

Intuitively we believed that DiGi board would perform better due to fact that it runs faster and has greater resources, but as results indicate we were mistaken – low-end board with less resources outperformed high-end ARM board when operating as gateway performing data delivery from WSN to user PC in local network. Furthermore achieves this with less energy consumed. This is particularly important when gateway must work by using battery and possibly if application requires gateway to be mobile. Next step could be examination of mobility impact on gateway and WSN collaboration. Of course there are few factors that should be taken into account. First, delivery speed decreases if more data control – filtering, aggregation must be performed, because data handling time increases. Second, if distance from WSN to user increases, delivery speed most likely will decrease, because route increases and more network devices must be employed to send data.

V. Alternate Gateway Connection and Gateway Speed Deviation Tests

To make results more detailed and comparable to other settings we continued our test with:

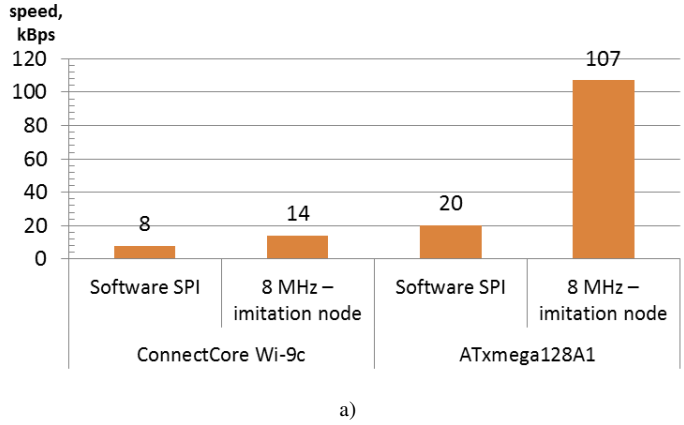
a) Transfer speed variation calculations for both connections;
b) Replacing SPI with UART, which is more common interface;

While transfer speed from sink node to user PC was measured, speed variations were calculated from min and max speed values from these observed results and depicted in Figure 5 a), b) and c). As can be seen in these three figures, ATXmega128A1 gateway variants very little and is even constant in some connections, but DiGi gateway variations highly greater than ATXmega GW. We think this is due to OS which needs to allocate/reallocate different resources which can introduce different and somewhat random delays in performance. This implies that DiGi can't be used for reliable (in sense of guaranteed data delivery time) communication because transfer speed can wary and guaranteed can be only lowest transfer speed. Thus when higher guaranteed data transfer rates should be guaranteed no OS should be used.

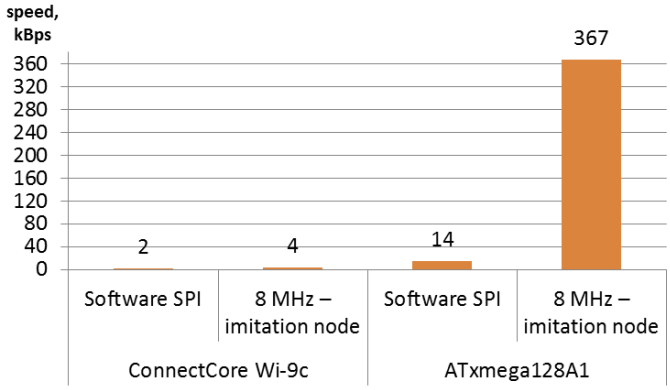
Since many sensor nodes use UART interface performance using this interface was measured as well. One drawback of UART in comparison with SPI is that UART frame has only 80% of useful data (8N1) in its frame while SPI has 100% - meaning only data is transferred. The same as in first test full communication link transfer speed between sink node and user PC were performed. Results are depicted in Figure 5 d). If compared with results from Figure 4 b) it is possible to see, that even ATXmega software SPI mode performs faster than UART. Although UART is more freely available, if transfer speed is important criterion it is advised not to use it.

A few lessons that were learnt during implementing presented GW were obtained. First, programming DiGi board was much faster and easier, because more examples are available for Linux, both for Ethernet and SPI. While ATXmega programming took a lot longer and debugging was more complicated. Second, even theoretically less powerful device can outperform high end device if certain conditions are met. Third, choosing gateway components can be difficult, but in the end it can present better performance as was seen in our case.

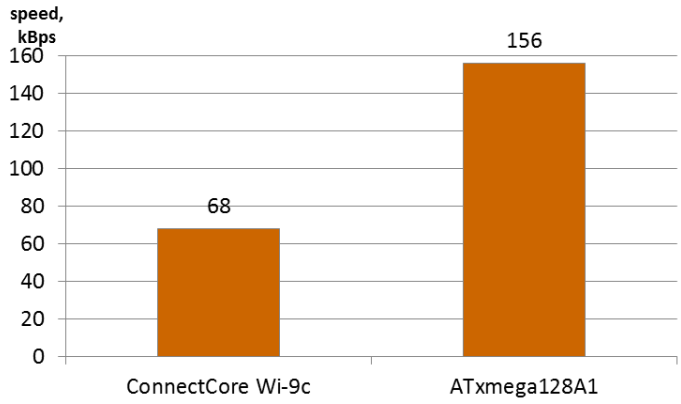
Our future work includes building a mobile gateway that could be used in wireless sensor network to relay local data to user and this research makes a contribution to the kind of board



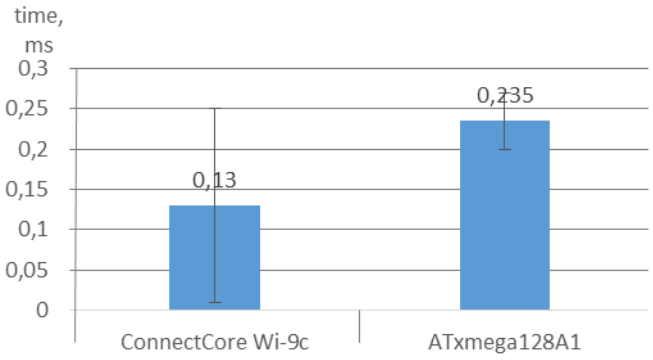
a)



b)

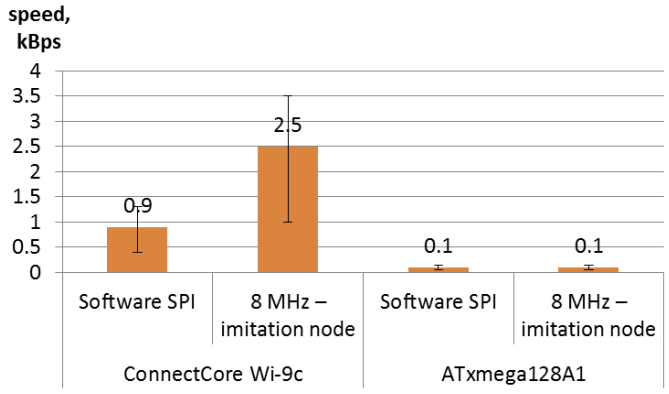


c)

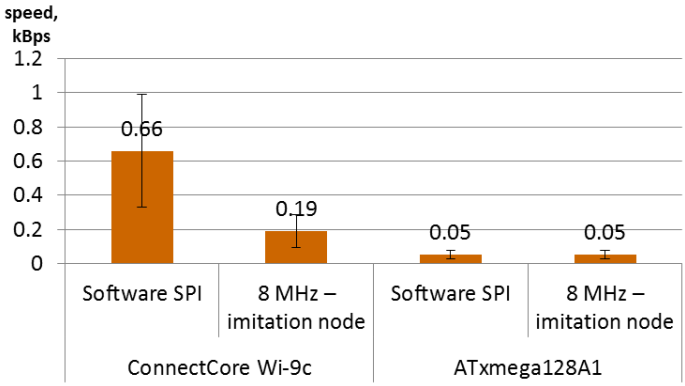


d)

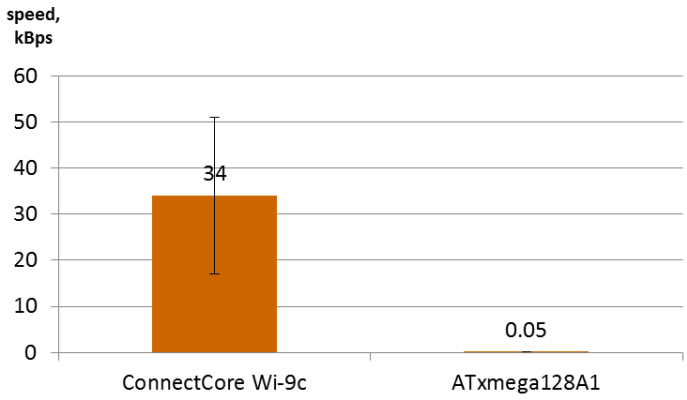
Fig. 4 Average transfer speed: a) from sink node to user PC; b) from gateway to sink node; c) from gateway to user PC; and d) average switching time between communication streams in gateways



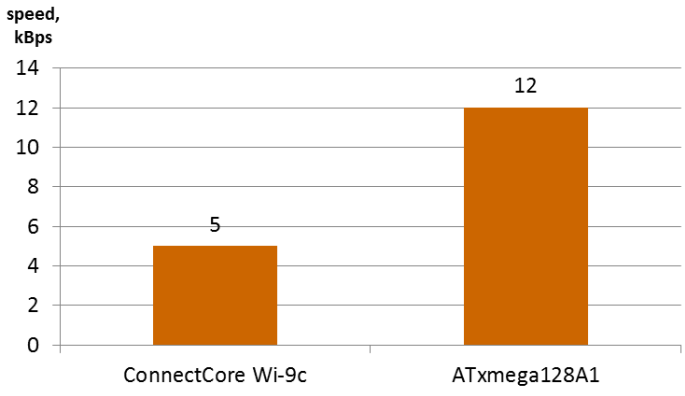
a)



b)



c)



d)

Fig. 5 Speed variation in a) sink node to user PC communication; b) sink node to gateway communication; c) gateway to user PC communication; and d) transfer rate from sink node to user PC using UART at 115200 baud rate

we should be using when implementing mobile gateway, i.e. we will consider using 8 bit microcontroller based gateway, since a lower energy consumption will provide longer operation. Most important resource for this type gateway is energy, although by using mobility gateway could be recharged more easily. Amongst new challenges could be adding wireless capabilities to this gateway.

VI. Conclusions

Our paper presents comparison of two different gateways envisioned for use in wireless sensor networks. Main concern that is investigated is choosing suitable hardware for gateway. Common approach, as we have seen in previous researches is to choose ARM based boards that run Linux OS. We suggested using less complex and less powerful board thus reducing energy consumption and furthermore data transfer rate shouldn't decrease significantly. Two available boards were chosen – ATXmega Xplained-A1 and DiGi Wi-9C board – and compared in similar test beds performing identical task – forwarding data from wireless sensor network sink node to user PC.

As obtained results imply using less powerful board without OS, can ensure smaller energy consumption and even increase data delivery speed, thus being more suited for wireless sensor network applications where data delivery speed is important. Furthermore, using the same board results are more stable over time, i.e. delivery speed is the same today as was yesterday. Further advantages of using less complex hardware is that overall costs for wireless sensor network can be reduced. Among disadvantages – programming becomes more complex when no API (which is in OS case) is used; possible that gateway must be designed by developer, because not always all necessary hardware is included in one board.

One more conclusion is from observing variations in results. Especially among sink node and GW when SPI is used. We observed that random delays between transferred data bytes was present. To our believes this is one of main reason why in the end ATXmega board outperformed DiGi board. Greatest drawback of OS based solutions is that to operate OS some user invisible processes are performed and some delays are introduced leading to uncertainty which is undesired in timely applications.

Lastly we want to mention some tradeoffs we encountered and observations we saw during developing our first gateways and give other developers some pointers what type of board would be more suited for certain applications:

1) If implementation should be done in short time preferable are boards with OS, like Linux. Because using API noticeably decreases development time. Furthermore using API provides wider application possibilities faster.
2) If gateway should ensure less energy consumption or little as possible speed deviation, or more control over hardware then board with no OS should be used (as seen in paper even less powerful board can be feasible).
3) If wireless sensor network costs are important then less powerful board can be used.

Acknowledgment

This work has been partly supported by the European Social Fund within the project «Support for the implementation of doctoral studies at Riga Technical University

References

[1] Lei, Shu and Jin, Wang and Hui, Xu and Cho, Jinsung and Lee, Sungyoung. Connecting Sensor Networks with TCP/IP Network. Springer-Verlag. p. 330--334 2006 http://dx.doi.org/10.1007/11610496_44

[2] Song, Ping and Chen, Chang and Li, Kejie and Sui, Li. The Design and Realization of Embedded Gateway Based on WSN. IEEE Computer Society. p. 32--36 2008 http://dx.doi.org/10.1109/CSSE.2008.889

[3] Guinard, Dominique and Trifa, Vlad and Pham, Thomas and Liechti, Olivier. Towards Physical Mashups in the Web of Things. IEEE Press. p. 196--199 2009 http://dl.acm.org/citation.cfm?id=1802340.1802386

[4] Han, Yanyan and Li, Deshi and Chen, Jian and Wang, Tianyu. Research on Wireless Sensor Network and Carrying Network Integration Based on Gateway. IEEE Computer Society. p. 748--751 2011 http://dx.doi.org/10.1109/iThings/CPSCom.2011.18

[5] Ye, Dun-fan and Min, Liang-liang and Wang, Wei. Design and Implementation of Wireless Sensor Network Gateway Based on Environmental Monitoring. IEEE Computer Society. p. 289--292 2009 http://dx.doi.org/10.1109/ESIAT.2009.194

[6] Shakya, Mukesh and Zhang, Jianhua and Zhang, Ping and Lampe, Mattias. Design and Optimization of Wireless Sensor Network with Mobile Gateway. IEEE Computer Society. p. 415--420 2007 http://dx.doi.org/10.1109/AINAW.2007.146

[7] Zhu, Qian and Wang, Ruicong and Chen, Qi and Liu, Yan and Qin, Weijun. IOT Gateway: BridgingWireless Sensor Networks into Internet of Things. IEEE Computer Society. p. 347--352 2010 http://dx.doi.org/10.1109/EUC.2010.58

[8] Atzori, Luigi and Iera, Antonio and Morabito, Giacomo. The Internet of Things: A Survey. Elsevier North-Holland, Inc.. p. 2787--2805 2010 http://dx.doi.org/10.1016/j.comnet.2010.05.010

[9] eZ430-RF2500 Development Tool: User's Guide, Texas Instruments Inc., Dallas, TX, 2009