Accelerated Connection Establishment (ACE) Mechanism for Bluetooth Low Energy

Konstantin Mikhaylov
Centre for Wireless Communications
University of Oulu, P.O. BOX 4500, Oulu, Finland
Email: konstantin.mikhaylov@ee.oulu.fi

***Abstract*—In the paper we focus on the problem of improving the efficiency of the peer-to-peer connection establishment for the recently suggested Bluetooth Low Energy (BLE) protocol. We suggest a novel Accelerated Connection Establishment (ACE) mechanism which enables to significantly reduce the time and energy consumption for establishing the connection between BLE devices in the multi-node scenarios. We discuss in details the background of the problem, introduce the suggested solution, and present the results of BLE network simulations, which confirm the efficiency of the suggested mechanism.**

***Index Terms*—Bluetooth Low Energy, Accelerated Connection Establishment, Bluetooth Smart, BLE, ACE, Network, Energy, Time**

I. Introduction

Bluetooth Low Energy (BLE) [1] is a perspective energy-efficient short-range wireless communication protocol, which was developed for enabling low power consumption, low complexity and low-cost wireless communication for consumer devices [1], [2]. The previous studies have proved that the BLE technology is potentially capable of providing higher throughput (see e.g., [3]–[5]) and is more energy efficient (see [5]–[9]) than such state-of-art technologies as ZigBee. Another important advantage of the protocol is that already today there are plenty commercial mobile phones, tablets and computers (called " Bluetooth Smart Ready" devices [10]) which integrate the dual-mode radio transceivers supporting both the communication over classical Bluetooth and BLE. This enables easy connectivity of the BLE-based sensors and the Bluetooth Smart Ready devices, and might enable multiple novel exciting applications thus playing important role in bringing the Internet of Things (IoT) concept to life.

Although the first version of the standard has been introduced in 2010, the technology got serious attention from the academy quite recently. E.g., the BLE technology has been used for developing the applications reported in [11]–[15]. The performance of the protocol and of the currently available commercial BLE transceivers were discussed in [3]–[9]. Nonetheless, in all these works, the authors were investigating the peer-to-peer (P2P) BLE links, leaving aside the multi-node scenarios. Among the major reasons for this are: a) the complexity of the BLE communication, which impedes its analysis and b) the absence of the tools capable of simulating the BLE networks. In our previous work [16] we have presented the novel tool capable of simulating the behavior of the multi-node BLE networks and have identified two scenarios causing degradation of the performance in multi-node BLE networks. In the current study we focus on one of these scenarios and suggest the mechanism which significantly reduces the time and energy consumption for establishing the connection between BLE devices in the multi-node scenarios.

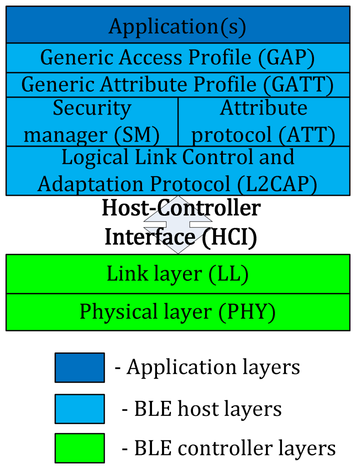


Fig. 1. BLE stack

II. Overview of BLE Technology and Problem Description

A. BLE Overview

The BLE protocol stack consists of the two major components: a BLE Controller and a BLE Host (see Fig. 1). Those two components might reside either on the same physical device or can be implemented by the different devices. The Controller is the logical entity that is responsible for the physical (PHY) and the link layer (LL). The Host implements the functionalities of the upper layers which we leave out of the discussion.

On the physical layer (PHY) BLE uses the Gaussian Frequency Shift Keying (GFSK) modulation with a bandwidth bit period product equal to 0.5 and the symbol rate of 1 mega-symbols per second. To simplify the transceiver design, BLE uses very short data packets with the maximum length of 47 bytes [5]. The power of output radio signal for BLE transmitters ranges from -20 to +10 dBm. The sensitivity

978-1-4799-4912-0/14/$31.00 © 2014 IEEE

level of the BLE receiver is below -70 dBm [1]. Likewise the previous version of Bluetooth, BLE operates in the license-free 2.4 GHz industrial, scientific and medical (ISM) band, which the protocol divides into 40 2-MHz wide channels. Three of those, which reside between the typically used wireless local area network channels, are called *advertising channels* and are used for advertising and service discovery. The remaining 37 *data channels* are used for establishing the P2P links between the devices. The data transmission of BLE devices is bound to time units known as advertising and connection events.

The advertising events might be used by the transceivers to broadcast small blocks of data, or to request and to specify the parameters of the connections to be established in the data channels. The period between the starts of two consecutive advertising events is defined as:

$$T_{advEvent} = advInterval + advDelay \tag{1}$$

where $advInterval$ is a integer multiple of 0.625 ms in the range of 20 ms to 10.24 s, and $advDelay$ is a pseudo-random value generated anew for each advertising event, which takes values in the range from 0 ms to 10 ms [1]. At the beginning of an advertising event the advertiser, i.e., the device which has some data to transmit, sends an advertising frame. Using the different advertising frame types, the advertiser might encapsulate up to 31 bytes of data straight in the advertising frame or might indicate its capability to start a connection in data channels. In the latter case, after sending a frame, the advertiser switches to receive and waits for possible connection establishment requests. If the connection request from a device (which is referred to as an initiator) is received, the two devices start peer-to-peer connection in the data channels. Depending on the specifics of the implementation and the requirements of the application, the BLE advertiser might either send the advertisements in a specific advertising channel or switch sequentially between few of those.

Once a connection in data channels is established, the *initiator* becomes the *master* and the *advertiser* becomes the *slave*. Note, that BLE assumes, that a master is more rich on resources than a slave. In the start of a connection event (referred to as the connection event anchor point), the used radio channel is changed following the pre-agreed sequence. The communication in each connection event is started by the master sending a frame to the slave. Then, the slave and the master alternate sending the frames on the data channel while at least one of the devices has data to transmit or until the current connection event ends. If either of devices receives two consecutive frames with CRC errors, the connection event is closed. The same happens if either of the devices misses a radio frame. The period between the frames on a data channel equals to the Interframe Space period (IFS) = 150 μs. The maximum LL payload of a BLE data frame is just 27 bytes [5], [17].

Once the connection event is closed, in order to save the energy, the devices might switch to a low-power sleep mode and wait until the start of the next connection event. The parameters of connection (e.g., the connection event interval

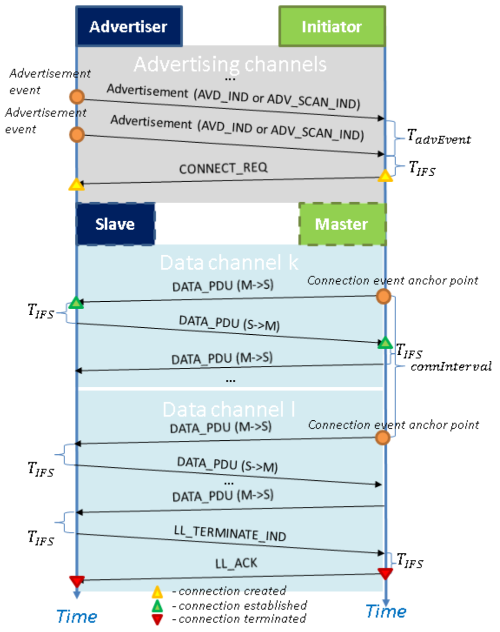


Fig. 2. Illustration of advertising, connection establishing, data transferring and connection terminating in BLE

- *connInterval* or the list of used data channels) are defined by the master and reported to the slave in the connection request (CONNECT_REQ) packet. Then, if required and if supported by both devices, *some* connection parameters (e.g., *connInterval*, *connSlaveLatency*, *supervisionTimeout*) might be modified without re-establishing the connection.

The timing of connection events is determined through two parameters, namely the *connInterval*, and the slave latency (*connSlaveLatency*). The *connInterval* is a multiple of 1.25 ms and has values ranging from 7.5 ms to 4.0 s. The *connSlaveLatency* ($connSlaveLatency \leq 500$) defines the maximum number of consecutive connection events in which a slave device is not required to listen.

The connection might be terminated at any time by either of the devices. Also the connection is terminated automatically on connection supervision timeout (*supervisionTimeout*, which ranges from 100 ms to 32 s).

The whole procedure including advertising, connection establishing, data transferring and connection terminating is illustrated in Fig. 2. Note, that neither on the advertising channels nor on the data channels the BLE devices do not use any sort of listen before talk mechanism.

B. Targeted Problem Description

In [16] we have presented the results of the simulations of multi-node BLE networks. The presented results reveal that in the case if multiple devices try to establish the connection at the same time, some of those might suffer from significant delays and energy losses, which increase dramatically with the increase of the used *connInterval*. After extensive studies of the nodes' behavior, we figured out that the major reason causing this, is the loss of the connection establishment request

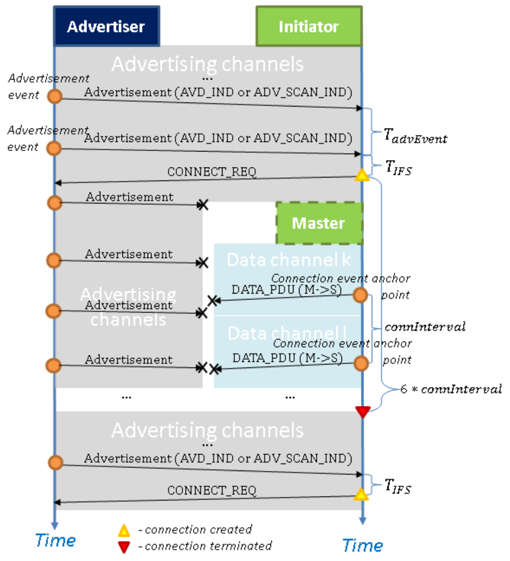


Fig. 3. Illustration of the CONNECT_REQ loss scenario

packets due to the mutual interferences.

As discussed in Section II-A and illustrated in Fig. 2, for establishing the connection the initiator sends a CONNECT_REQ packet, after receiving which both nodes switch to data channels. Now let's consider what happens if the advertiser misses CONNECT_REQ - see Fig. 3. According to the BLE specification [1], after sending the CONNECT_REQ the initiator considers the connection to be successfully created and switches to the first data channel, which is derived from the *hopIncrement* value and the *channelMap* sent in CONNECT_REQ. After sending the CONNECT_REQ, the master (i.e., the former initiator) arms the supervision timeout after 6**connInterval* and attempts to reach the slave in the data channels. Meanwhile, the advertiser, which has not got the CONNECT_REQ, continues in the advertising channels. Obviously, as both nodes are now operating in different frequency channels, those are unable to hear each other. Once the supervision timeout expires, the master returns to the advertising channels and might try re-establishing the connection. As easy to see, for the case of high *connInterval* values ($20ms \leq connInterval \leq 10.24s$), this takes quite significant time. The other problem related to the discussed scenario is the energy consumption of the advertiser. As discussed in Section II-A, in each advertising event a connectible advertiser needs to send a frame and then to listen for a possible CONNECT_REQ on *each* of the used advertising channels. Consequently, in the case if $connInterval > advInterval$, the energy consumption of the resource-poor advertiser (i.e., the slave) might be significantly higher compared with the one of the resource-rich master.

As easy to see, for the discussed problem coming into existence, there should be another node transmitting in the same advertising channel when the initiator sends its CONNECT_REQ. Although this is not very likely if all BLE devices in the network start advertising at random moment of time and do not return to the advertising channels once the connection is successfully established, for the network with event-triggered data transfers the problem might arise much more often. E.g., let's consider the case of m BLE sensors and multiple gateways located in the same area. Let's assume, that the sensors do not transmit any data if the sensed parameter is below the specified threshold and issue an alarm, once the sensed parameter exceeds the threshold. If all the sensors detect the event and initialize the advertising at the same moment of time, have the same $advInterval$ and start advertising in the same channel (which is very likely given that the sensors are identical), the probability of having a collision during the connection establishment phase is given by:

$$P_{cont}(m) = \begin{cases} 0 \text{ if } m = 1 \\ 1 \text{ if } 1/(m-1) \geq T_{ADV}/T_{rand} \\ 1-(1-(m-1)T_{ADV}/10ms)^m \text{else} \end{cases} \quad (2)$$

where $T_{ADV} = 630 + 8 \cdot n \;\; \mu s$ is the time required for establishing the connection (i.e., total time for sending ADV packet, IFS, and receiving the CONNECT_REQ; n stands for the payload of the ADV packet, $n \in [0, 31]$ for ADV_IND or $n = 6$ for ADV_DIRECT_IND), and $T_{rand} = max(advDelay) + T_{ADV}$. Note, that some of the current real-life BLE transceivers instead of generating a random $advDelay$ just randomly pick one of the values from the limited set of pre-defined values (e.g., [18]). In this case, the probability of collision is given by:

$$P_{discrete}(m) = \begin{cases} 1 \text{ if } m \geq D \\ 1 - D!/(D^m \cdot (D-m)!) \quad \text{else} \end{cases} \quad (3)$$

where $D \leq 10$ is the number of $advDelay$ values in the discrete set. The effect of the BLE sensors number on the probability of having at least one packet collision during connection initialization for the described scenario is depicted in Fig. 4. As easy to see, in the case if there are more than four devices, the probability of having at least one collision exceeds 0.5 and for more than 10 nodes - a collision is almost inevitable. Although not each and every collision will bring the described problem to life, this is quite possible.

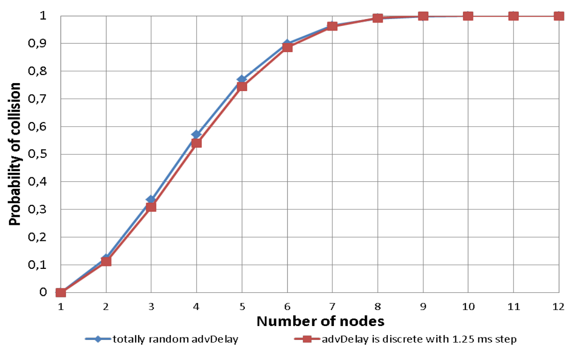


Fig. 4. Probability of having at least one collision during connection establishment in the network of BLE sensors with event-triggered data transfer (n=6, D=9)

III. Accelerated Connection Establishment (ACE) Mechanism

To handle the discussed problem we suggest to use a double-stage connection establishment mechanism named Accelerated Connection Establishment (ACE) which uses different values for the *connInterval* on each stage. Instead of setting the desired *connInterval* value straight ahead in the CONNECT_REQ packet, we propose to use at the initialization phase very small values of *connInterval*. In the case of CONNECT_REQ loss, this enables to significantly reduce the connection supervision timeout for the master device and thus enables the master to get back in the advertising channels much faster. Meanwhile, if the connection is established successfully (i.e., if the master gets a packet from the slave in the data channels), the master should use the BLE connection parameter update mechanisms for changing the *connInterval* to the desired value.

Note, that for the real-life BLE transceiver, the described mechanism might be implemented in two ways: either at the LL of the Controller or at the logical link control and adaptation protocol (L2CAP) of the Host.

In the former case, instead of using the *connInterval* provided by the Host in the LE_Create_Connection command, the LL sends the CONNECT_REQ with a small *connInterval*. Once successfully getting the first data frame from the slave, the LL of the master issues LL_CONECTION_UPDATE_REQ and modifies the *connInterval* to the one which was originally requested by the Host in the LE_Create_Connection command. Then, once the new *connInterval* is taken in use, the LL informs the Host about the successful connection establishment using LE_Connection_Complete_Event message.

In the latter case, the Host on the master device issues LE_Create_Connection command with small *connInterval* already set. Once getting the LE_Connection_Complete_Event message from the LL (which is issued once master has sent the CONNECT_REQ packet), the Host issues the LE_Connection_Update command and modifies the *connInterval* to the desired value. Note, that according to the BLE specification ([1], p. 2549), the master should allow a minimum of 6 connection events that the slave will be listening before the updated parameters will be taken in use. Therefore, in the case if the CONNECT_REQ was missed by the advertiser, the connection supervision timer of the master always fires earlier than the new *connInterval* is taken in use.

In both these cases, compared to the single-staged connection establishment procedure, the suggested mechanism implies the transmission of only one additional packet - a LL_CONECTION_UPDATE_REQ of 22 bytes which takes 176 μs.

IV. Evaluation of ACE

To check the performance of ACE, we have modified the BLE simulation tool presented in [16]. We have extended the simulator by adding the model of the network level which supports the most important BLE HCI commands and events. Also we implemented the BLE LL mechanisms which enable to change the parameters of BLE connections using LL_CONECTION_UPDATE_REQ and LL_CHANNEL_MAP_REQ packets. The ACE mechanism was implemented at the network level, using the second solution described in Section III.

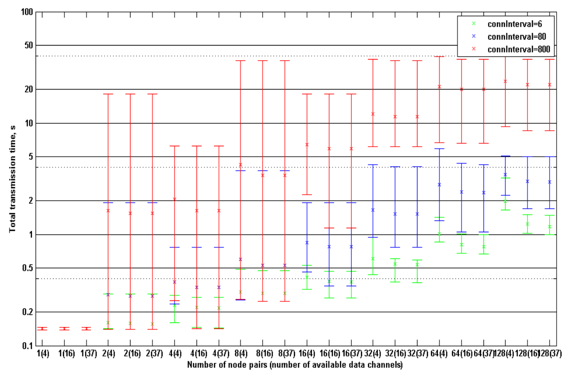
For testing the ACE, we simulated a network consisting of 1 to 128 pairs of BLE nodes randomly placed over the area of 10 by 10 meters. In the beginning of the simulation on each slave we placed 1000 bytes of data, which a slave should transfer to its master. All the nodes started at the same time in the advertisement channels and used those for establishing the connection (for this, ADV_DIRECT_IND events with *advInterval*=100 ms were used). Then the nodes switched to the data channels for the actual data transferring. After finishing the data transfer, the devices terminate the connection and enter sleep mode. The experiment ended once all the slaves have successfully transferred their data. For advertising, the nodes were using all three advertising channels.

The simulations were executed for the different *connInterval* and the different numbers of the used data channels for the two cases: first - when the desired *connInterval* was set straight ahead, and second - when ACE was used. For each set of parameters, the simulations were repeated for 25 different network layouts. The results revealing the time for transferring all the data and the energy consumption of the slave nodes ('x' denotes the mean value, the minimum and maximum values are shown as well) are presented in Fig. 5.

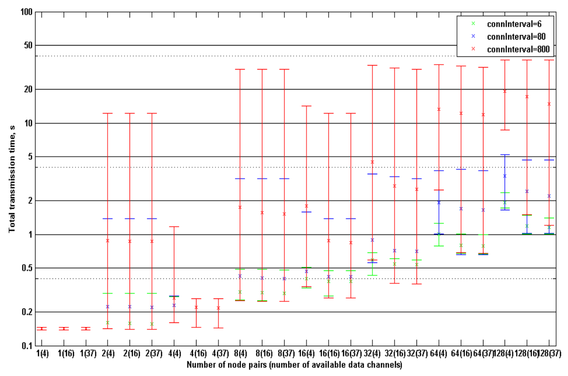
The presented results reveal that for high desired *connInterval*, for all tested multi-node BLE networks the suggested ACE significantly reduced both the time and the energy consumption for data transfer. In many cases (see, e.g., 32 pairs and *connInterval=800* - in average, the use of ACE reduced the required time by 63% and the consumed energy by 78%) the difference between the results obtained for the case without ACE and with ACE was in the order of times. For the case when just a single pair of BLE devices was active, the use of ACE slightly increased the required time and the energy consumption. Nonetheless, this increase was well below 1%. As one can see from the presented results, the use of ACE is especially beneficial for the multi-node networks with nodes having high desired *connInterval* values.

V. Conclusions

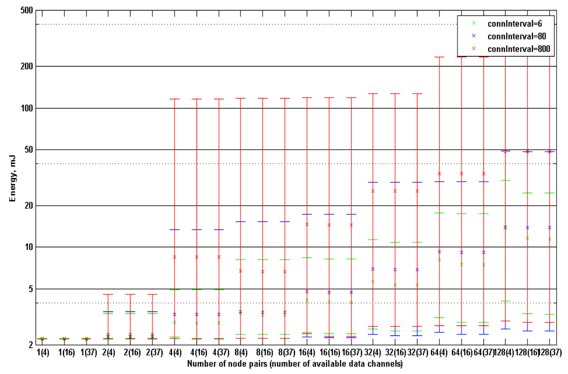
In the paper we focused on the problem of improving the efficiency of the connection establishment for the recently suggested Bluetooth Low Energy (BLE) protocol and suggested the Accelerated Connection Establishment (ACE) mechanism. Despite its simplicity, ACE enables to significantly reduce the time and energy consumption for establishing the connection between BLE devices in the multi-node environment. Although for the single-node case the use of ACE might slightly increase the energy consumption, we have shown that this increase is minor and is hardly comparable to the benefits which the mechanism provides for the multi-node case.



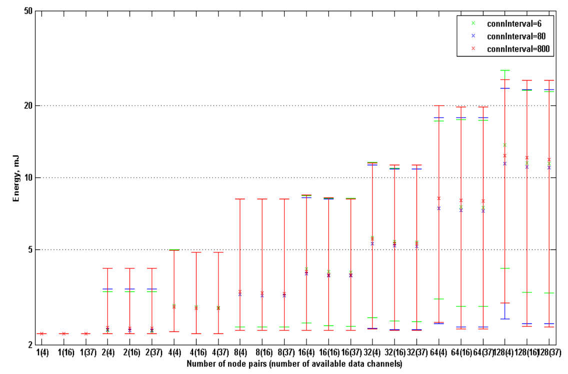
(a) Data transfer time without ACE



(b) Data transfer time with ACE



(c) Slaves' energy consumption without ACE



(d) Slaves' energy consumption with ACE

Fig. 5. Simulation results

We have shown that the suggested mechanism might be easily implemented using the standard means of BLE protocol and might be incorporated either to the link layer (LL) of the BLE controller or to the logical link control and adaptation protocol (L2CAP) of the BLE host.

References

[1] *Bluetooth Specification version 4.1*, The Bluetooth Special Interest Group, Kirkland, WA, USA, 2013.

[2] R. Heydon, *Bluetooth Low Energy: the developers handbook*. Upper Saddle River, NJ: Prentice Hall PTR, 2012.

[3] C. Gomez et.al.,"Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-power Wireless Technology," *Sensors*, vol. 12, no. 9, pp. 11734-11753, Aug. 2012.

[4] C. Gomez et.al., "Modeling the Maximum Throughput of Bluetooth Low Energy in an Error-Prone Link," *IEEE Commun. Lett.*, vol. 15, no.11, pp. 1187-1189, Sept. 2011.

[5] K. Mikhaylov et.al., "Performance analysis and comparison of Bluetooth Low Energy with IEEE 802.15.4 and SimpliciTI," *J. Sens. Actuator Networks*, vol. 2, no. 3, pp. 589-613, Aug. 2013.

[6] M. Siekkinen et.al., "How Low Energy is Bluetooth Low Energy? Comparative Measurements with ZigBee / 802.15.4," *Proc. IEEE Wireless Commun. and Networking Conf. Workshops*, Paris, France, 2012, pp. 232-237.

[7] M.C. Ekstrom et.al., "A Bluetooth Radio Energy Consumption Model for Low-Duty-Cycle Applications," *IEEE Trans. Instrum. Meas.*, vol.61, no.3, pp. 609-617, Mar. 2012.

[8] E. Mackensen et.al., "Bluetooth Low Energy (BLE) based wireless sensors," *Proc. IEEE SENSORS*, Taipei, Taiwan, 2012, pp. 1-4.

[9] A. Dementyev et.al., "Power consumption analysis of Bluetooth Low Energy, ZigBee and ANT sensor nodes in a cyclic sleep scenario," *Proc. IEEE Int. Wireless Symp.*, Beijing, China, 2013, pp. 1-4.

[10] *Bluetooth Smart and Smart Ready products now available* (26.02.2014), Available: http://www.bluetooth.com/Pages/Bluetooth-Smart-Devices-List.aspx

[11] B. Yu et.al., "Bluetooth Low Energy (BLE) Based Mobile Electrocardiogram Monitoring System," *Proc. Int. Conf. Inf. and Automation*, Shenyang, China, 2012, pp. 763-767.

[12] M. Al et.al., "A Bluetooth Low Energy Implantable Glucose Monitoring System," *Proc. 8th European Radar Conf.*, Manchester, UK, 2011, pp. 377-380.

[13] A. J. Jara et.al., "Evaluation of Bluetooth Low Energy Capabilities for Continuous Data Transmission from a Wearable Electrocardiogram," *Proc. 6th Int. Conf. Innovative Mobile and Internet Services Ubiquitous Computing*, Palermo, Italy, 2012, pp. 912-917.

[14] Y. Park and H.Cho, "Transmission of ECG data with the patch-type ECG sensor system using Bluetooth Low Energy," *Proc. Int. Conf. ICT Convergence*, Jeju, ROK, 2013, pp. 289-294.

[15] K. Zidek, and J. Pitel, "Smart 3D pointing device based on MEMS sensor and Bluetooth Low Energy," *Proc. IEEE Symp. Computational Intelligence Control and Automation*, Singapore, Singapore, 2013, pp. 207-211.

[16] K. Mikhaylov, "Simulation of Network-Level Performance for Bluetooth Low Energy," to be presented at *25th Annu. Int. Symp. Personal, Indoor, Mobile Radio Commun., Washington*, DC, USA, 2014.

[17] K. Mikhaylov and J. Tervonen, "Multihop Data Transfer Service for Bluetooth Low Energy," *Proc. 13th Int. Conf. ITS Telecommunications*, Tampere, Finland, 2013, pp. 319-324.

[18] *Bluetooth low energy software stack and tools* (24.06.2014), Texas Instruments, Available:http://www.ti.com/tool/ble-stack