Plug-and-play mechanism for plain transducers with wired digital interfaces attached to wireless sensor network nodes

Konstantin Mikhaylov*, Tomi Pitkäaho and
Jouni Tervonen

RFMedia Laboratory,
Oulu Southern Institute,
University of Oulu,
Vierimaantie 5, 84100, Ylivieska, Finland
Email: konstantin.mikhaylov@oulu.fi
Email: tomi.pitkaaho@oulu.fi
Email: jouni.tervonen@oulu.fi
*Corresponding author

Abstract: The ability to connect sensors to the Wireless Sensor Network (WSN) nodes without the need for physical device configuration has many advantages: application development is simplified, network deployment and service is easier, and sensors can be swapped or added on-the-fly. The existing solution for sensor Plug-and-Play (P&P) for WSN nodes is the IEEE 1451 set of standards developed for smart transducers. The serious drawback of this solution is that it cannot be used with the most widespread plain transducers without adding multiple external components. Therefore, in this paper, we introduce a novel mechanism that allows implementation of P&P connection to WSN nodes for commercially available off-the-shelf sensors with the most widespread wired plain digital interfaces (SPI, I2C, 1-wire etc.) without any single external component utilisation.

Keywords: plug-and-play mechanism; wireless sensor networks; plain transducers with wired digital interfaces; sensor identification; sensor discovery; digital sensor; WSN; P&P; I2C; SPI; 1-wire.

Reference to this paper should be made as follows: Mikhaylov, K., Pitkäaho, T. and Tervonen, J. (2013) 'Plug-and-play mechanism for plain transducers with wired digital interfaces attached to wireless sensor network nodes', *Int. J. Sensor Networks*, Vol. 14, No. 1, pp.50–63.

Biographical notes: Konstantin Mikhaylov has been working as a Researcher for Wireless and Embedded Systems since 2008 at the RFMedia Laboratory of Oulu Southern Institute, the University of Oulu, Ylivieska, Finland. He received his BSc (2006) and MSc (2008) in Electrical Engineering, with the focus on wireless systems, from St. Petersburg State Polytechnical University, St. Petersburg, Russian Federation. His main research interests include embedded systems, non-specific short range wireless communication technologies and wireless sensor networks, especially the problems of energy efficiency, available resources estimation and hardware-resources aware operation adaptation for those systems.

Tomi Pitkäaho received the MSc degree in Information Processing Science from the University of Oulu, Finland, in 2008. He worked as a Research Assistant from 2006 to 2008 and as a Researcher from 2008 to 2010 in Wireless Data Transmission research group in the RFMedia Laboratory of Oulu Southern Institute, University of Oulu. From 2010 to 2011 he worked as a Project Researcher and since 2011 as a Project Manager in the Digital Holography research group in the RFMedia Laboratory. His current research interests include digital holography, digital holographic microscopy, image processing, 3D image processing, and digital hologram image processing.

Jouni Tervonen has been working as the Research Manager with Oulu Southern Institute, the University of Oulu, since 2004. He received his Dipl Eng (MSc) and DSc (Tech) in Electrical Engineering from the Helsinki University of Technology, Espoo, Finland in 1992 and 1997, respectively. He is in charge of the Wireless Data Transmission and Digital Media Groups of Oulu Southern Institute at the RFMedia Laboratory in Ylivieska, Finland. Currently, his main research interests are wireless sensor networks, embedded systems, and applications of emerging 4G telecommunication standards.

Copyright © 2013 Inderscience Enterprises Ltd.

1 Introduction

The technological advances that have occurred during recent years have extended the range of the parameters that can be measured using existing transducers. Today's sensor technologies combine high precision, fast operation, and low energy consumption within relatively small casings, which has made possible the rapid development and dissemination of various Distributed Measurement and Control (DMC) applications. As revealed in the work of Akyildiz et al. (2002), Bertocco et al. (2008), Chee-Yee and Kumar (2003) and Yunseop et al. (2008) the use of wireless communication between the sensor nodes reduces installation and maintenance costs and provides high levels of network scalability and flexibility, making Wireless Sensor Networks (WSNs) one of the key technologies for the future. As shown in the work of Akyildiz et al. (2002), Akyildiz and Xudong (2005) and Yunseop et al. (2008), contemporary WSNs are often implemented as mesh networks with dynamic self-organisation and self-configuration, which simplifies DMC application deployment and service.

At the core of a WSN node are the actual sensors (see Figure 1) that make the WSN meaningful. However connecting these general sensors usually requires modifications to the node's hardware, or at least to the node's software, which makes the nodes application-dependent. Connection of different sensors without device modifications now relies on a Plug-and-Play (P&P) approach that has been standardised as part of the Institute of Electrical and Electronics Engineers (IEEE) IEEE-1451 set of standards for intelligent smart sensors interfaces (Gilsinn and Lee, 2001; IEEE Std. 1451.0, 2007; IEEE Std. 21 450, 2010; Lee et al., 2004; Wobschall, 2008). The availability of sensor P&P (according to Gumudavelli et al. (2010), a sensor can be considered P&P if it becomes operational and networked after turned on and is physically connected to a WSN node's microcontroller) provides several significant benefits for WSNs, such as:

- simplified application development, device manufacturing, WSN deployment, and service (when manufactured, a WSN node does not require any sensor-specific software – it can be obtained, e.g. from the WSN once node is switched on and its sensors are identified);
- deployed WSN nodes can be upgraded or dynamically reassigned for new tasks by changing the sensors;
- sensors that disconnect for unexpected reasons from a WSN node can be automatically taken out of use.

The general concept of a smart transducer, developed in the late 1980s, is that it is a device that combines both a sensing system and a local microcontroller with required interface circuitry, processor, memory, and a circuitry for implementing network communication (Song and Lee, 2008a). According to Lee and Song (2005) and Song and Lee (2008b), the smart transducers also implement system level functions (e.g. measurement compensation, automatic calibration, self-diagnosis) and networking communication functions such as node identification and node loss detection. The IEEE 1451 set of standards defines a common communication interface for connecting these types of smart transducers to digital systems and instruments in network-independent environments. The standard also defines the hardware interfaces for connecting transducers to a microcontroller or to an instrumentation system, and the set of software interfaces for connecting transducers to a network (Lee and Song, 2005; Potter, 2002). One of the IEEE 1451 key elements is the definition of the Transducer Electronic Data Sheet (TEDS) format. The TEDS are the memory blocks that are embedded into each sensor (see Figure 2) and contain information about the sensor's manufacturer, model, serial number, measurement range, sensitivity, and various calibration information, all of which can be used in sensor self-identification and self-description as discussed in the work of Potter (2002) and Ross et al. (2009).

Figure 1 Structure of a common WSN node

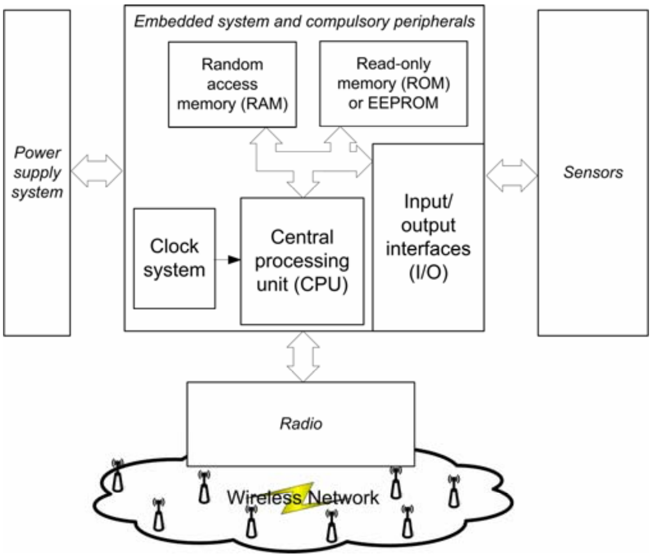
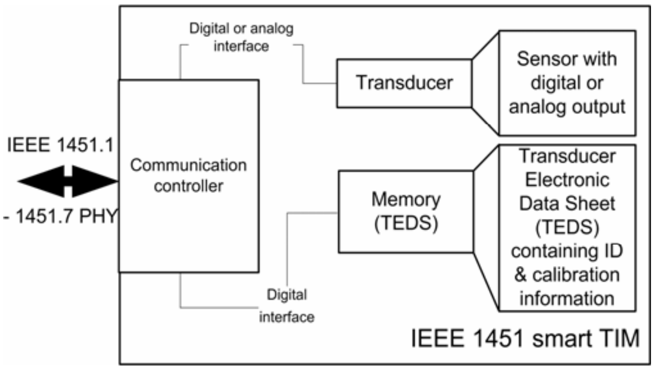


Figure 2 TEDS mechanism in IEEE 1451

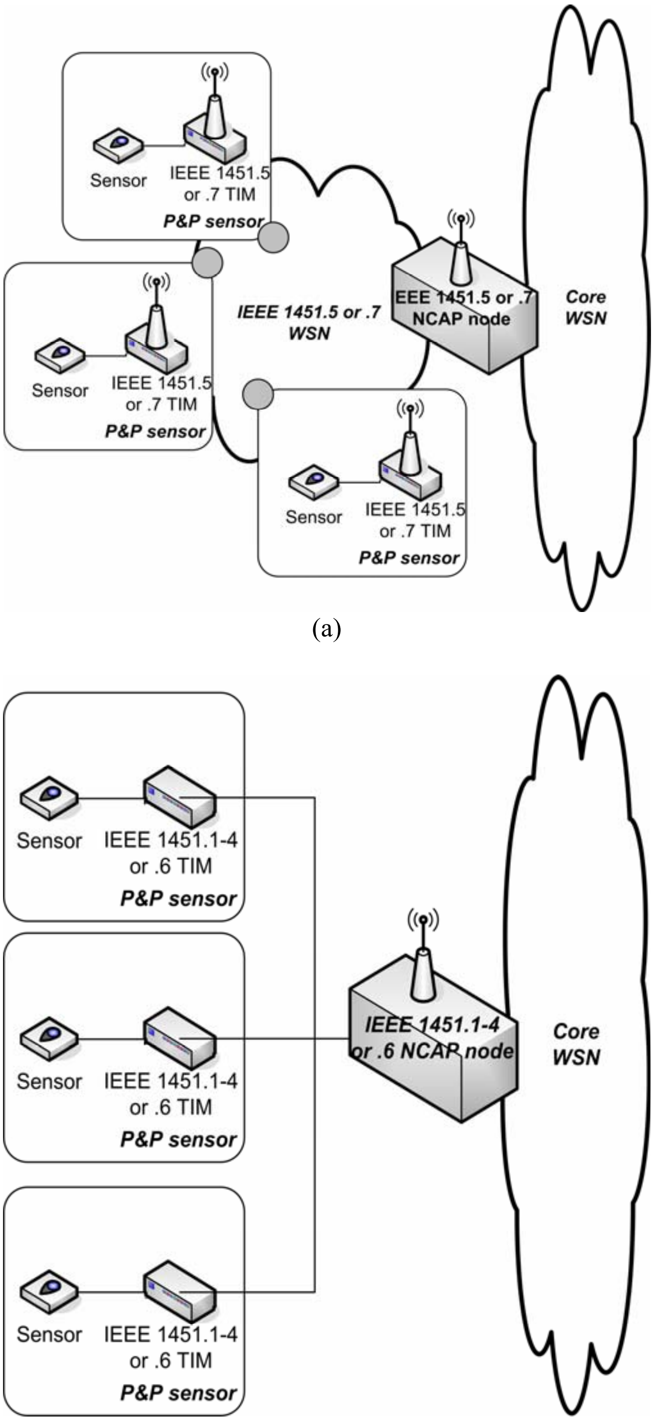


Sensor P&P for WSN nodes using the IEEE 1451 can be implemented in two ways. The actual sensors can be wirelessly connected using IEEE 1451.5 or .7 Transducer Interface Modules (TIMs) to the Network Capable Application Processor (NCAP), which will provide further connection to the core WSN (Figure 3(a)). Alternatively, Gilsinn and Lee (2001) and Wobschall (2008) suggested, multiple sensors can be connected over IEEE 1451.1-4 or .6 wired interfaces to the NCAP, which is connected to the WSN (Figure 3(b)). Both of these solutions need to use IEEE 1451 TIMs between the actual sensors and the WSN node (NCAP in this case). Provision of the minimum required functionality requires that these TIMs contain a memory chip for storing the TEDS, an appropriate multiplexing circuitry for separating the TEDS and the transducer data, and a required communication interface controller (e.g. for IEEE 1451.1-1451.7 physical (PHY) layers) (see Figure 2). An implementation of additional IEEE 1451 features often requires the use of a separate microcontroller or a processor on the TIM (consider, e.g. Lee et al. (2004) and Ross et al. (2009)). Needless to say, these can significantly increase the price and power consumption of the resulting P&P sensors, which is especially undesirable for WSN.

Although today's WSNs and even a single WSN node can have substantial intelligence, the majority of the sensors that are currently used on WSN nodes are still very simple devices that do not support any smart features (see Figure 1 for typical structure of a WSN node) (Ovalle et al., 2010). The use of these simple sensors allows reduction in the cost and power consumption of the WSN nodes, which is important due to restricted resource availability of many WSN applications. Nonetheless, as noticed in the work of Kuorilehto et al. (2007) and Dunbar (2001), the absence of smart features within these sensors restricts the implementation of P&P sensor connections to a WSN node.

The IEEE 1451 cannot be used with plain transducers without significant hardware modification and use of new components. Therefore, in this paper, we introduce a novel P&P mechanism intended for the Commercially available Off-The-Shelf (COTS) sensors with the most widespread (according to Yurish, 2012 and Avnet, 2012) plain wired digital interfaces (namely – Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I^2C) interface, 1-wire and proprietary ones) connected to a WSN node. The suggested P&P mechanism is not based on the smart sensor concept and this allows us to dispense with all of the components between the sensor and the WSN node (compared, e.g. to Figure 3(b)); thus, it reduces the price and increases the applicability of the solution. The developed mechanism uses currently existing WSN node resources, and the resources within a WSN to implement a P&P support for the sensors. The introduced mechanism can be used as a less expensive and simpler alternative to the IEEE 1451 for implementing the sensor P&P connection to WSN nodes. In this paper, specifics of communication in WSN are not addressed and the WSN nodes are assumed to have the required mechanisms already in place for secure and reliable data transmission within the network.

Figure 3 Wireless transducer plug-and-play implementation for WSNs using IEEE 1451, (a) Wireless P&P sensor connection to a NCAP (over IEEE 1451.5 or .7 interfaces) with a wireless interface between the NCAP and the core WSN (b) Wired P&P sensor connection to a NCAP (over IEEE 1451.1-4 or .6 interfaces) with a wireless interface between the NCAP and the core WSN



(a)

(b)

The remainder of the article is organised as follows: Section 2 describes the suggested P&P mechanism, including the mechanisms for detection of sensor connection/disconnection to/from a WSN node, sensor identification, and retrieval of P&P support data from WSN. Section 3 presents the results of the P&P mechanism implementation and a real-life evaluation. Section 4 concludes the paper and summarises the results.

2 Suggested sensor plug-and-play mechanism for WSN

The suggested sensor P&P mechanism for plain sensors connected to a WSN node includes four major operation stages:

1. Detection of sensor connection and disconnection;
2. Identification (ID) data retrieval from WSN;
3. ID of connected sensor(s);
4. Software driver retrieval for the identified sensors from a WSN.

Therefore, its implementation requires the three following mechanisms:

A Sensor connection/disconnection detection;

B Connected sensor identification;

C Mechanism for retrieval of the required data from the WSN (both the ID data and the microcontroller program code to be used with the new sensors (sensor driver)).

2.1 Sensor connection/disconnection detection

Smart sensors can announce their connection to a WSN node's microcontroller, whereas plain sensors usually do not have this capability. The main reason for this is that the plain sensors with digital interfaces are implemented as slave devices for appropriate buses (e.g. SPI, I^2C) (see NXP Semiconductors (2007) and Motorola Semiconductor (2003)). This requires that all of the communication with these devices must be initialised by the master device (for WSNs nodes, this will be a microcontroller or other controlling device of the WSN node (see Figure 1)). Therefore, an external mechanism is needed to inform the WSN node's microcontroller that its peripherals have been changed. We suggest using the following four mechanisms for the sensor connection/disconnection detection by WSN node:

1. External signal;
2. Radio command;
3. Periodic identification launching of connected devices;
4. New device connection detection using the WSN node's power consumption monitoring.

The external signal usage assumes that when a sensor is connected or disconnected, this is the result of an intentional external impact. During this impact, the controls of the node could also be accessed, which can be used to inform the node about a sensor change. This can be implemented, e.g.

- by equipping the sensor node with a button or switch that will be activated each time the sensor is changed;
- by rebooting the sensor node (the sensor discovery should be launched automatically after each reboot)
- by using a special design of P&P sensor interface (e.g. it can include a wire that will be connected on the sensor board to the ground or the power supply line thus signalising that the sensor has been attached).

Another possibility for WSN nodes is to use a radio command to inform the node that its peripherals have been changed. In this case, after service operations that involve sensor changes, the node should receive a radio command that will trigger the sensor identification procedure. This command can be issued by the network access point or by the special nodes that are used during the service operations.

The third option is a periodic launch of the device identification subroutine. A disadvantage of this method is its overhead processing due to inability to get actual information about the new sensor connection to the WSN node. This method also can cause significant delays between the actual sensor connection and detection and initiation of a new device.

The fourth option is sensor connection detection based on the power consumption of the WSN node. Clearly, the addition or removal of a sensor also influences the overall node power consumption. The main difficulty in implementing this method is that many sensors, whenever they are not in use, switch automatically to a low-power mode with very low power consumption and thus become difficult to detect.

In the real system, combination of several of these sensor connection detection methods is possible, depending on the required characteristics and available resources.

The detection of a disconnection of a digital sensor is straightforward – if for some reason one of the sensors gets disconnected from the WSN node, the node will not get a reply from this sensor while trying to communicate with it.

2.2 Connected sensors identification

Once the WSN node's microcontroller recognises the presence of new sensors, it should identify them. According to Avnet (2012), the most widespread digital interfaces (e.g. for temperature sensors) are I^2C (57% of devices), SPI (10%), and 1-wire (6%) devices; the rest of the sensors utilise company-specific digital interfaces. Unfortunately, of the most widely utilised sensor digital interfaces, only the 1-wire interface has a mechanism for a single-valued sensor identification (see e.g. Maxim Integrated Products, 2002). Among the rest interfaces, only the I^2C has some support for sensor identification based on the sensor's 7-bit address (suggested by Ptasinski and Sassi (2002)). However, the single-valued identification using the (suggested by Ptasinski and Sassi (2002)) mechanism is impossible because the addresses for I^2C sensors are not unique and multiple I^2C sensors can use same address (NXP Semiconductors, 2007) (see Appendix A for an example). The other interfaces, such as SPI or the majority of the company-specific ones, have no identification mechanism available at all (Motorola Semiconductor, 2003).

Our suggested solution is to use the existing features of plain digital interfaces (e.g. the address mechanism available for I^2C devices) and the features of the sensors themselves (e.g. the data within the sensor's registers). This solution can be implemented by using a simple table-based trial-and-error algorithm, which is executed by the WSN node's microcontroller to which the sensors are connected (see Figure 1). The suggested algorithm (see Figure 4) utilises a prefilled ID table – the table containing unique ID request and expected ID reply data for each sensor, which can be potentially connected to a WSN node. Depending on the P&P sensor specifics, as ID request can be used a single command or a set of commands sending which to the P&P sensor of this type will generate the specific unique ID reply. By going through this ID table (see Figure 4), sending the specified ID requests and comparing the obtained P&P sensor's replies with the expected ones, a WSN node's microcontroller detects which of the sensors in the ID table are attached to it. At the ID stage, a WSN node's microcontroller does not need to have a complete driver for controlling a new sensor; instead it uses only a minimum set of commands that are required to obtain the reply from a sensor, which significantly limits the memory consumption for the ID algorithm. As already discussed, some plain digital interfaces do not have standard ID mechanisms and the devices do not include special ID information — for identification of these, we suggest using for ID request one or combination of several of the following four methods:

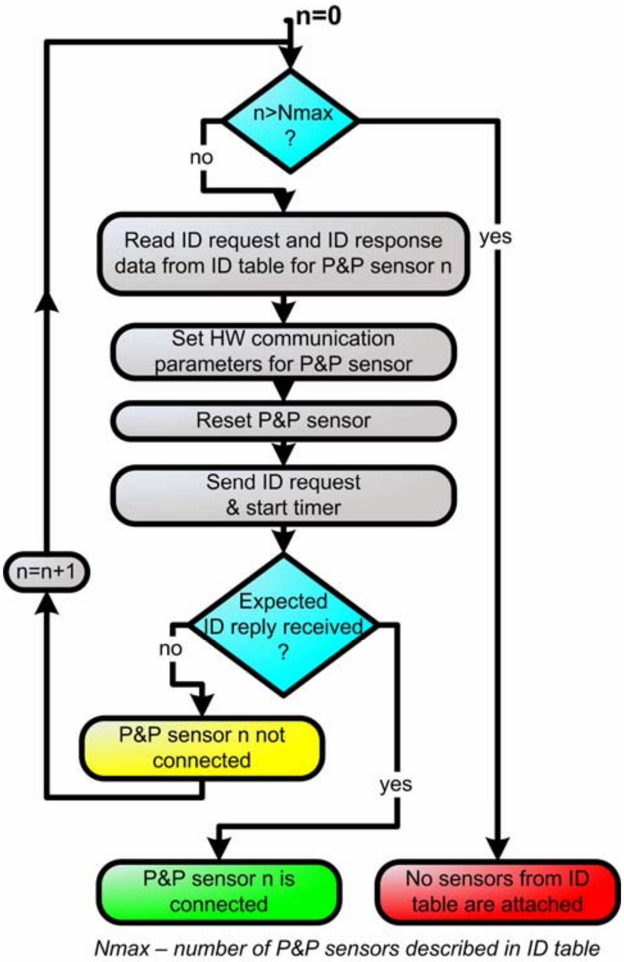
1 Read from the ID registers or any other registers that contain known-in-advance data (this device identification is based on the facts of hardware connection settings correctness, register/command existence, and retrieved data correctness).

2 Sequentially write to and read from a register with inaccessible bits (bits containing a value which cannot be changed) (device identification is based on the facts of hardware connection settings correctness, register existence, and inaccessible bit locations).

3 Execute a command for which the range of possible return values is known (e.g. make a temperature measurement) (device identification is based on the facts of hardware connection settings correctness, command execution acknowledgement, and returned data falling within known limits).

4 Sequentially write to and read from certain registers or certain command execution (device identification is based only on the facts of hardware connection setting correctness and register existence/command execution acknowledgement).

Figure 4 Suggested sensor identification algorithm based on an ID table tryout (assuming that only a single sensor can be connected to an interface) (see online version for colours)



The suggested mechanism assumes that each plain P&P sensor has a unique ID request-response sequence that can be constructed using four suggested above methods. Although the validity of this assumption for all of the available sensors is impossible to confirm, the material presented in Section 3 shows that the probability of two sensors having exactly same data in the same registers is rather low. However, in the case where two or more sensors do have absolutely the same data in all of the registers, the identification of these would not be possible using the suggested method. In that case, the identification data for the next stage (i.e. the driver retrieval) can be provided either manually (e.g. by sending the special radio packet containing the ID for attached sensor) or by using more complicated techniques (e.g. by trying out the drivers for all identical sensors and comparing the obtained value with the data from sensors of the same type on near-by WSN nodes).

The required ID information and the connection features for implementing sensor P&P over the most widespread digital sensor interfaces are presented in Tables 1 and 2, respectively.

Table 1 Required ID information for devices with I^{2}C, SPI, 1-wire and proprietary digital buses

Data	*I^{2}C device*	*SPI device*	*1-wire device*	*Device with proprietary bus*
Clock	required	required	not required	required
Addresses	required[a]	not required	required	depend on bus
ID request:				
send data	required	required	required[b]	required
delays	required	required	required	required
service operations	not required	not required	not required	required
ID response	required	required	not required[b]	required

Notes: [a] Can have several different addresses depending on connection.

[b] Uses only the fact of response, thus only one command that generates a response is required, the actual response data are not important.

2.3 ID data and driver storage and retrieval

As well known (Kuorilehto et al., 2007; Mohammadi and Jadidoleslamy, 2011; Rajkamal and Ranjan, 2011), WSN nodes often have rather limited resources. Among these is the available memory, which complicates storage of the ID information and the program code for working with all potentially attachable sensors on each WSN node. Luckily, the networking capability of a WSN can be used to solve this problem. We suggest keeping only the program code for the actual connected sensors on the WSN node during normal operation, while the required ID data and the software drivers for potentially connectable sensors are stored elsewhere in the network 'resource centre' (RC). The main function of a RC is to provide the necessary ID data and the software drivers, by request (see Figure 5), to the end-device (ED) WSN nodes. A network access point (AP) or a separate node can be used as a RC (e.g. RC1 on Figure 5). As one of the options, the data for P&P implementation can be stored on remote location (e.g. on the internet) and the RC can act as an access point to this remote location using the appropriate communication technologies (e.g. RC2 on Figure 5). The network can have several RCs, but in this case, an appropriate access mechanism should be implemented. If a WSN node is not capable of storing the entire ID table, the table can be divided in several parts that the ED will sequentially download from the RC and go through. These solutions allow to handle the problem of ID table scalability. Updated versions of the ID table and the sensor drivers for EDs are also provided by the RC nodes. Needless to say, prior to connection of any new P&P sensor to a WSN node, the ID information for it should be included in ID table.

Table 2 P&P properties for devices using I^{2}C, SPI, 1-wire and proprietary digital buses

Data	*I^{2}C device*	*SPI device*	*1-wire device*	*Device with proprietary bus*
Physical connection[a]	specified	specified	specified	depend on bus
Number of physical lines	2(2)[b]	2(2)[b] + 1 per device	1(2)[b]	depend on bus
Possible sensor ID methods[c]	-physical connection -address -unique request/reply	-physical connection -unique request/reply	-physical connection -unique address	-depend on bus
Sensor connection detection method	see II-A	see II-A + using CS line	see II-A	depend on bus

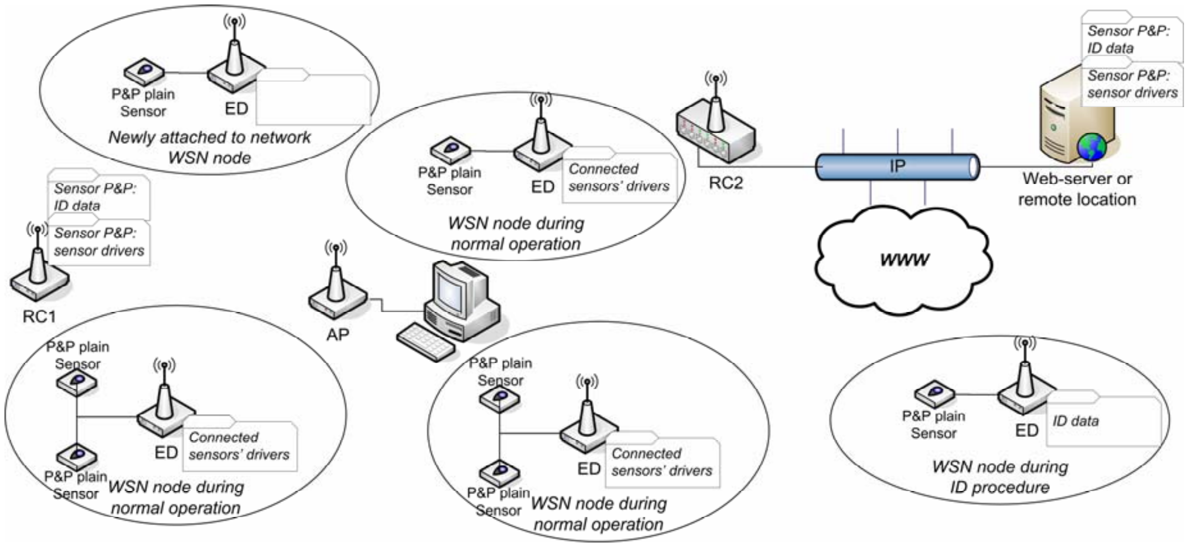
Notes: [a] Sensor connection (pinout) is predefined

[b] N (M) where N – number of communication lines, M – number of supply lines

[c] For factors used for device identification, see Section 2.2

The implementation of the suggested approach obviously requires capability for secure and reliable data transmission between and ED and RC. In the current paper, we set aside the implementation of this and assume that it is already provided by the WSN (see e.g. Yi et al. (2007) and Yun et al. (2008)) through utilisation of error-detecting, error correcting and packet retransmission mechanisms.

Figure 5 The structure of a WSN with on-node sensor P&P (ED – end device, AP – access point, RC – resource centre)



3 Implementation and evaluation for the developed sensor plug-and-play method

The suggested plain sensor P&P mechanism was evaluated in two phases. First, the features of the currently existing plain digital sensors were analysed and the suggested sensor identification algorithm operation was simulated based on the information of 48 existing I^{2}C devices. For the second phase, the suggested mechanisms were implemented and evaluated with hardware using several existing I^{2}C sensors and WSN

During the first phase of evaluation, we randomly chose 48 different real-life I^{2}C devices (of these, 42 were sensors and six were other devices). Based on the information provided in the device datasheets, we manually generated an ID table that could be used to distinguish these devices using the suggested ID algorithm (the ID table and the list of the tested devices can be found in the Appendix A). In addition, the further analysis of the device datasheets revealed that an ‘averaged’ I^{2}C device uses five different I^{2}C bus addresses and has 35 bytes of data in its registers, of which six bytes contain (after reset) either non-zero information or have some inaccessible bits.

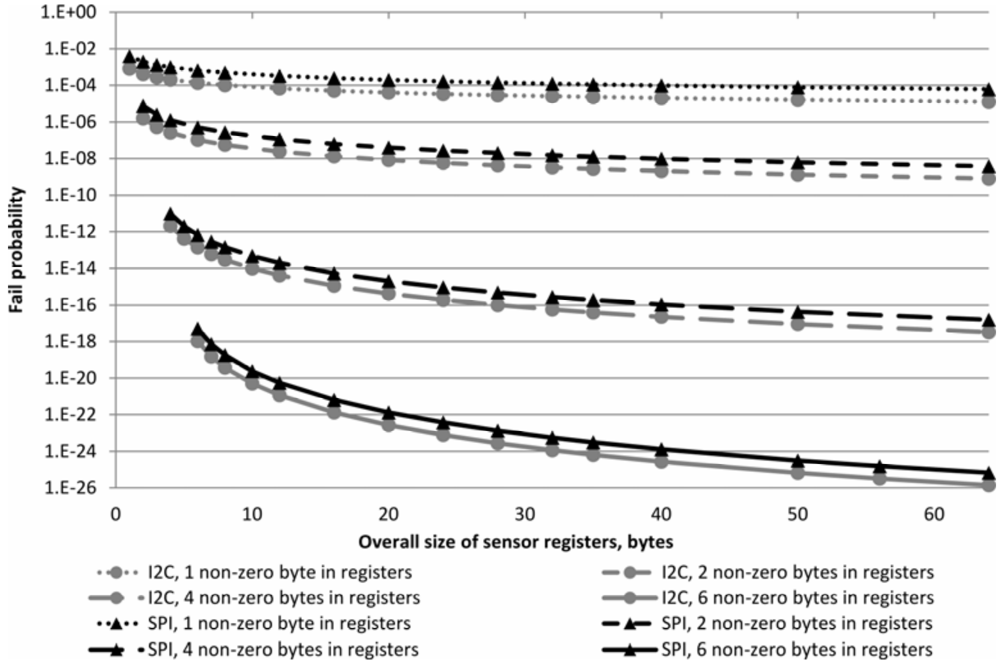
For estimating the applicability of the suggested solution, we have calculated the probability for two devices having at least one matching I^{2}C address and the same data in all non-zero registers (assuming that non-zero register addresses and data are random), which would make the suggested sensor ID mechanism inapplicable (see equation (1)).

$$P_{match} = \left(1 - \prod_{k=0}^{n_{addr}-1} \frac{N_{addr} - n_{addr} - k}{N_{addr} - k} \right) \times \left(\prod_{k=0}^{n_{bytes}-1} \frac{1}{N_{bytes} - k} \right) \times \left(\frac{1}{2^8 - 1} \right)^{n_{bytes}} \tag{1}$$

In equation (1), the Naddr is the overall number of possible device addresses (e.g. 112 for I2C bus, one for SPI), naddr – is the number of addresses one device can use (e.g. five for ‘average’ I2C device, one for SPI), Nbytes is the overall number of registers on a device, and nbytes is the number of non-zero registers on a device after reset. The resulting curves, showing the probability of the existence of two devices that cannot be identified using the suggested ID algorithm for I2C and SPI interfaces with different numbers of registers on the device, are presented in Figure 6. As shown in Figure 6, when the devices have at least two non-zero registers (which is true for more than 95% of the examined sensors), the probability of having two identical devices becomes lower than 10–5 and for an ‘averaged’ I2C device with five non-zero registers out of 32 registers, this probability reaches 10–23.

The ID table reveals (see the Appendix A) that, among the examined devices, only three (and of these, only one sensor) have no non-zero registers or registers with inaccessible bits at all after reset. Nonetheless, the suggested ID algorithm appears to be applicable even for these devices – these devices have been identified by writing some data to their registers and reading it back (the other examined devices with matching I2C addresses are unable to do this). In addition, three pairs of the examined devices appeared to have identical data in their registers. All of these devices were different modifications of the same sensor and had almost the same functionality and an identical set of commands. However, even these devices can be distinguished at a later stage by a proper driver implementation (the identification can be done using the returned measurement data – e.g. for LSM320DL registers, 0x2A and 0x2B do not contain measurement data as they have only a 2D gyroscope, while for LSM330DL, some data will be present as it encapsulates a 3D gyroscope).

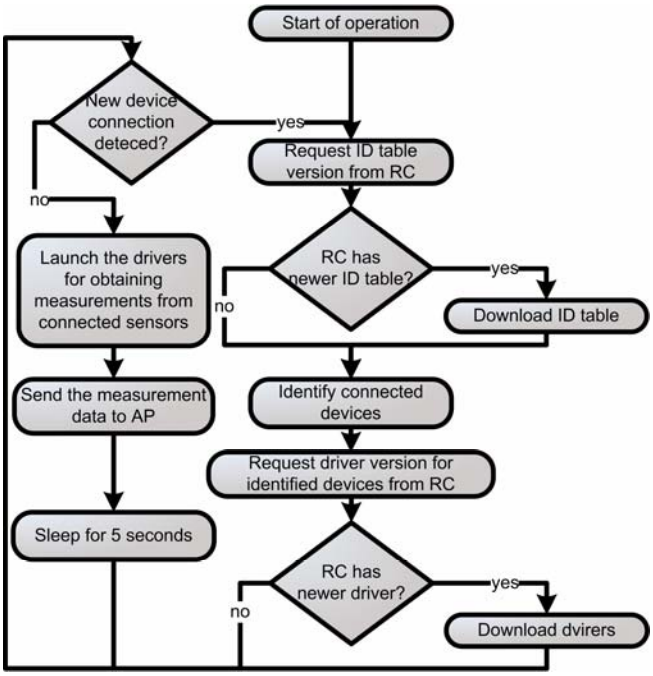
Figure 6 Probability for two devices not distinguishable by the suggested P&P mechanism



We estimated the complexity and evaluated the suggested sensor P&P mechanism in real-life during the second phase by realising a hardware implementation. For this, we used Texas Instruments' (TI) EZ430-RF2500 (Texas Instruments, 2009) development boards, which have an on-board MSP430F2274 16-bit microcontroller (Texas Instruments, 2010) and a CC2500 2.4 GHz radio (Texas Instruments, 2011). The suggested P&P mechanism was implemented in full (i.e. the mechanisms for detection of I^2C device connection, retrieval of ID data from a resource centre, identification of the connected I^2C device and software driver retrieval were realised – see Figure 7 for the WSN node operation algorithm) for three plain I^2C devices, using the REBOS (Mikhaylov and Tervonen, 2011) operation system on the WSN node microcontrollers. The tested boards with P&P I^2C devices were directly connected to a microcontroller of the WSN through simple 4-pin connector (see Figure 8(a)) using 4 cm long wires without using any external components. We implemented connection detection of a new I^2C device by the WSN node by using the first two suggested options in Section 2.1 and the connected device was identified using the first of suggested methods in Section 2.2. Both the required ID information (i.e. the ID table) and the drivers during the test were stored on a special node (RC1 on Figure 8(b)) and access to it was implemented using an over-the-air reprogramming mechanism suggested in the work of Mikhaylov and Tervonen (2010) (before connecting to the WSN, the ED nodes *had no ID data or sensor drivers at all*). Since the main focal point of hardware evaluation was real-life testing of the suggested discovery, ID, and data retrieval mechanisms, we used a simple WSN with a single-hop star network topology (see Figure 8(b)) and a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism for wireless channel access in our tests. In the implemented WSN, the RC1 node acted as both the resource centre for the suggested P&P mechanism and the access point to the WSN network. The task of the laptop connected to the RC1 was to monitor the network operation and to update the ID and driver data on the RC, if required. We ensured an errorless ID table and driver retrieval by the WSN nodes by implementing a two-stage error detection mechanism (a Cyclic Redundancy Check (CRC) for the received packet using a radio and checksum control by a microcontroller) with full packet retransmission in the case of error at any stage.

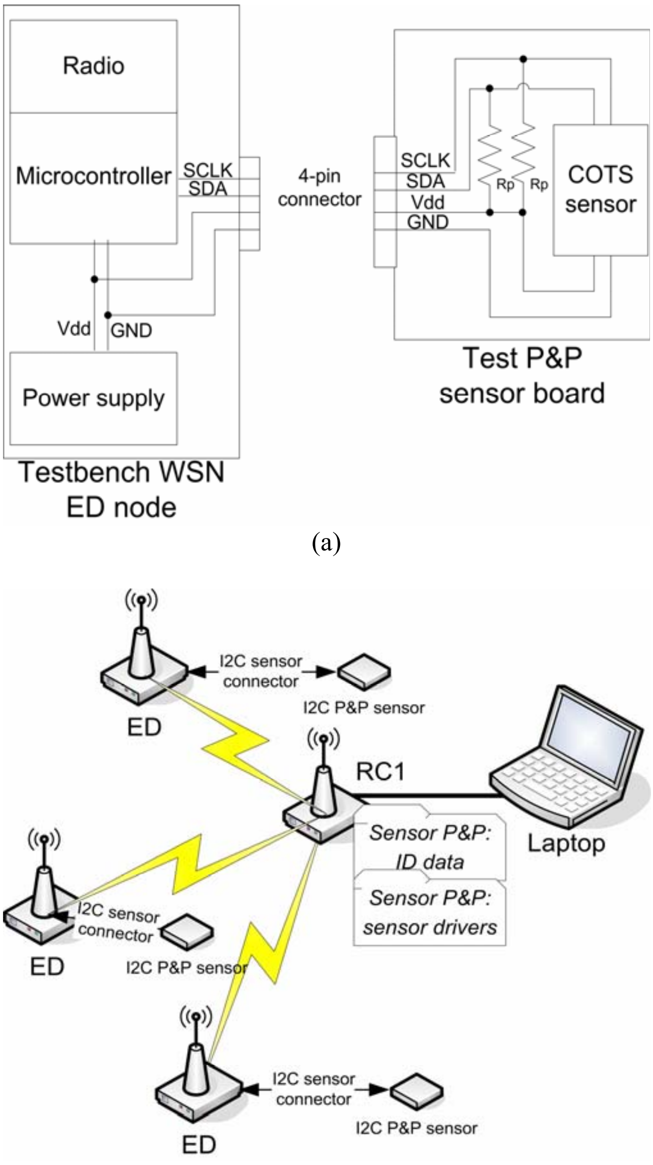
As shown in Tables 3 and 4, the implementation of the suggested P&P mechanism resulted in rather moderate resource consumption. As revealed in Table 3, the overall size of the code, which realises the suggested P&P mechanism in full (including the microcontroller's Operation System (OS) and the code for networking within WSN), is slightly above 10 kbytes (around 30% of the memory available on the microcontroller) and is not dependent of the number of possible I^2C devices in the ID table. The overall size of the ID table for the three I^2C devices used was 42 bytes (see the Table 4; an additional five bytes were required for the table header). As can be seen from the Table 4, the average size of the ID data for a single I^2C device is around 13 bytes, while for the 'averaged' I^2C sensor discussed above (a device with five possible addresses and six non-zero registers) the size of the ID data would reach 30 bytes for the worst case scenario, which allows storage of the ID table on a WSN node (using Mikhaylov and Tervonen's approach (2010)) simultaneously for at least 600 I^2C sensors. The size of sensor drivers – the microcontroller specific code that implements the minimum required I^2C sensor functionality (calibrates the sensor, orders it to make the measurement, processes the measurement, converts it to International System of Units (SI) and forwards the data to the application layer) – was 165 bytes for the TI TMP102, 179 bytes for the ST MPR121, and 233 bytes for the ST STMPE801.

Figure 7 Algorithm for the WSN node (ED) operation during P&P hardware testing



The effect of the suggested P&P mechanism on the WSN node operation is shown by the values for energy and time consumption, as well as the inbound and outbound data traffic during different stages for the execution of the sensor P&P mechanism are presented in Table 5 and Figure 9. These measurements represent the case when the WSN node microcontroller was running at a 1 MHz clock frequency with a power supply of 3.6 V and no errors occurred in the radio communication. The data in Table 5 are presented for the two contrasting scenarios: when the ED has just started and requires to download both the ID table and the drivers from RC (e.g. a new device attached to WSN); and for the opposite case, when the ED already had the latest versions of the ID data and required drivers (e.g. one of the sensors on the attached node has been removed).

Figure 8 I^{2}C Testbed, (a) Schematics of connection for a P&P I2C plain sensor to a WSN node (b) Topology of the network used for testing the suggested P&P mechanism



(a)

(b)

Table 3 WSN node microcontroller code and RAM consumption for suggested P&P algorithm implementation

Program stage	*ED Code size, bytes*	*ED RAM size[b], bytes*	*RC Code size, bytes*	*RC RAM size[b], bytes*
OS core + required drivers[a]	4676	160	3850	129
ID table request /response	1912	8	2294[c]	5
Connection detection and ID	1872	6	–	–
Driver request/response	1852	9	2294[c]	5
Normal work	550	6	–	–

Notes: [a] ID data and device drivers not included, their size can be found in Table 4

[b] Temporary allocated memory from stack not included

[c] ID and driver request processing has been implemented as single function, full function size is calculated.

Table 4 Structure and size of ID data and drivers (in bytes) for tested I^{2}C devices

Required data	*TI TMP102*	*ST MPR121*	*ST MPE801*
Device ID	1	1	1
ID Data length	1	1	1
Clock	1	1	1
I^{2}C Addresses (overall)	3	5	5
Number of I^{2}C addresses	1	1	1
Possible I^{2}C addresses	2	4	4
ID request (overall)	2	2	2
Data length	1	1	1
Data	1	1	1
ID response (overall)	4	3	4
Data length	1	1	1
Data	3	2	3
Overall ID data size	12	13	14
Overall driver size:[a]	165	179	233

Note: [a]Developed drivers provide the minimum required functionality.

Figure 9 Energy and time consumption distribution between different operations for ED with different sensors (both ID data and device drivers were downloaded from RC)

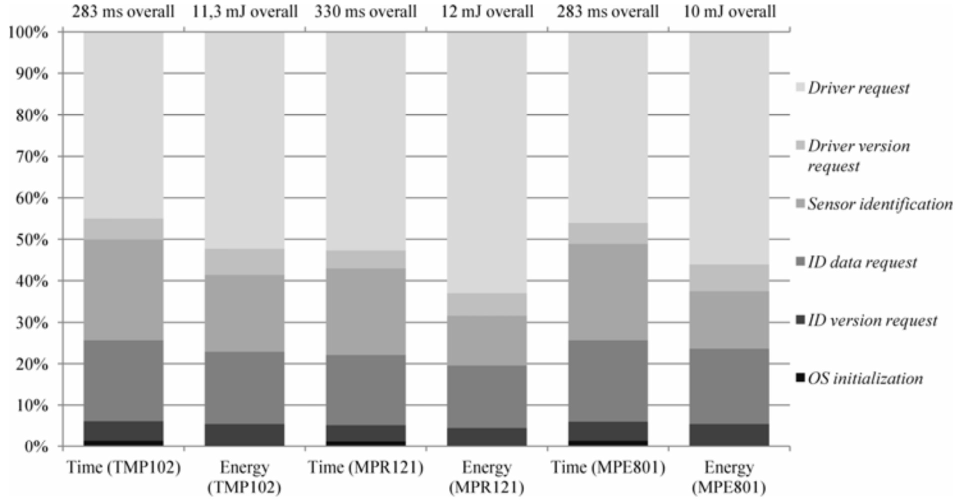


Table 5 Time and energy consumption and amount of radio traffic for implemented I^{2}C sensor P&P during different stages

Required resources	*TI TMP102*	*ST MPR121*	*ST MPE801*
Initialisation:			
Required time, ms	3.9	3.9	3.9
Power consumption, μJ	33.6	25.1	25.1
ID table request:			
Data transmitted, bytes	23/5[a,c] (7/1)[b,c]	23/5[a,c] (7/1)[b,c]	23/5[a,c] (7/1)[b,c]
Data received, bytes	68/44[a,c] (8/2)[b,c]	68/44[a,c] (8/2)[b,c]	68/44[a,c] (8/2)[b,c]
Required time, ms	69.5[a] 13.5[b]	69.9[a] 13.3[b]	69.7[a] 13.3[b]
Power consumption, μJ	2560[a] 581.8[b]	2334.2[a] 515.1[b]	2337.5[a] 515.6[b]
Devices detection and identification:			
Required time, ms	69.6	69.5	69.7
Power consumption, μJ	2093.6	1441	1378.9
Driver request:			
Data transmitted, bytes	53/12[a,c] (8/2)[b,c]	71/16[a,c] (8/2)[b,c]	53/12[a,c] (8/2)[b,c]
Data received, bytes	221/165[a,c] (9/3)[b,c]	309/233[a,c] (9/3)[b,c]	235/179[a,c] (9/3)[b,c]
Required time, ms	143[a] 14.5[b]	190[a] 14.6[b]	147[a] 14.6[b]
Power consumption, μJ	6635.9[a] 712.9[b]	8245.2[a] 659.1[b]	6254.3[a] 652.7[b]
Overall:			
Data transmitted, bytes	76/17[a,c] (15/3)[b,c]	94/21[a,c] (15/3)[b,c]	76/17[a,c] (15/3)[b,c]
Data received, bytes	289/209[a,c] (17/5)[b,c]	377/277[a,c] (17/5)[b,c]	303/223[a,c] (17/5)[b,c]
Required time, ms	283[a] 101.5[b]	330[a] 101.5[b]	283[a] 98.5[b]
Power consumption, μJ	11308[a] 3425[b]	12034[a] 2640[b]	9970[a] 2573[b]

Notes: [a] Driver and ID data downloading required

[b] No driver or ID data downloading required

[c] Overall data including wireless protocol and service data/actual I^{2}C ID or driver data

4 Discussion and conclusion

The Plug-and-Play connection of sensors to the WSN nodes allows simplification of the application development, network deployment and further service procedures, and provides on-the-fly sensor changing capability. The reported implementations for sensor P&P connection to the WSN nodes usually use IEEE 1451 smart sensors, which have multiple useful features (e.g. self-identification, automatic calibration, self-diagnosis), but are rather complicated and thus expensive devices that are not really widespread on the current market. However, the majority of COTS plain digital transducers that are widely used on WSN nodes do not have any mechanisms for single-valued identification.

Therefore, in the current paper, we have suggested a novel P&P mechanism for COTS plain digital transducers utilising most widespread wired digital serial interfaces connected to the WSN nodes. The suggested mechanism is a complete solution for sensor P&P and allows the WSN nodes:

- to detect the connection of a new sensor;
- to retrieve the required data for sensor identification from the WSN;
- to identify the connected sensors (using a simple table-based try-out algorithm and the specifics of sensors' architecture);
- to retrieve the software code for using the identified sensor from the WSN.

The suggested P&P mechanism is not limited to any specific physical, data link, network or transport layers protocols of WSN and can be used with any protocol that ensures reliable transmission of P&P data in a WSN. Also, the suggested P&P mechanism involves the transmission of the service data only once the node with P&P sensor is attached to the WSN or once the sensors of the node are changed. This allows us to expect that, if the sensor changes will not happen too often, the suggested P&P mechanism will not have negative influence on the data traffic in the WSN.

The major advantage of the suggested mechanism is that it uses the resources of the *already existing* WSN node processing devices and the resources available in the WSN, which allows implementation of the suggested P&P mechanism with COTS plain digital sensors, and *without a single external component*. This is especially important for the WSNs with restricted resources. The major disadvantage of the suggested method, which is a consequence of the external component usage refusal, is that it is not always applicable for the simple sensors that do not have *any* data in their registers. Nonetheless, as has been shown during the evaluation phase, the number of these sensors is sufficiently small. Other significant disadvantages, which are consequences of the sensor identification approach used, are following: the ID table should include information about all of the sensors that can be connected to a WSN node *at any time*. For the current implementation, the size of data in ID table for each sensor is around 14 bytes. The amount of data traffic in the WSN and the time required for sensor identification *increases linearly* with the number of P&P sensors that can potentially be connected to the WSN nodes. Nonetheless, in many cases, the suggested P&P mechanism could provide a much *less expensive* and *simpler* alternative to smart sensors and the IEEE 1451 interface.

The simulations and hardware implementation of the suggested P&P mechanism using Texas Instruments eZ430-RF2500 boards and different I^{2}C sensors showed that the suggested mechanism has rather moderate resource requirements (the suggested P&P algorithm occupied less than 30% of the available microcontroller's memory). For the case of three different I^{2}C devices, the suggested method is able to provide sensor P&P connection within one third of a second. The comparison of the suggested mechanism with the reported implementations of the IEEE 1451 system (see e.g. Cummins et al. (1998) and Stepanenko et al. (2006)) shows that the suggested mechanism allows reduction of the required microcontroller code amount by more than 90%, while at the same time allowing exclusion of additional components (such as memory blocks for storing TEDS or additional processing devices for implementing additional smart features). In addition, the evaluation of the suggested P&P mechanism showed that it makes the initial WSN node program *independent of the connected sensors* – all of the required sensor drivers are *retrieved automatically* from the WSN by the node after start-up and peripheral identification.

Although we focused primarily on the sensors connected to the WSN nodes, as these are the most often used peripheral devices, the suggested P&P mechanism can be extended to a broad range of other peripheral devices that use standard wired digital buses (e.g. memory chips, ADC/DAC, pin extenders etc.). Likewise, the suggested mechanisms for device connection detection and device identification can also be applied to other embedded processors besides microcontrollers and other system besides WSNs.

As a further research we are planning to investigate the influence of proposed P&P mechanism on the power consumption and data flows within WSN consisting of multiple nodes for different scenarios and the networking and scalability issues within the WSN utilising suggested P&P mechanism.

References

Akyildiz, I., Weilian, S., Sankarasubramaniam, Y. and Cayirci, E. (2002) 'A survey on sensor networks', *IEEE Communication Magazine*, Vol. 40, No. 8, pp.102–114.

Akyildiz, I. and Xudong, W. (2005) 'A survey on wireless mesh networks', *IEEE Communication Magazine*, Vol. 43, No. 9, pp.S23–S30.

Avnet (2012) *Avnet Electronics Marketing Database.* Available online at: http://avnetexpress.avnet.com (accessed on 30 January 2012).

Bertocco, M., Gamba, G., Sona, A. and Vitturi, S. (2008) 'Experimental characterization of wireless sensor networks for industrial applications', *IEEE Transactions on Instrumentation and Measurements*, Vol. 57, No. 8, pp.1537–1546.

Chee-Yee, C. and Kumar, S. (2003) 'Sensor networks: evolution, opportunities, and challenges', *Proceedings of IEEE*, Vol. 91, No. 8, pp.1247–1256.

Cummins, T., Byrne, E., Brannick, D. and Dempsey, D. (1998) 'An IEEE 1451 standard transducer interface chip with 12-b ADC, two 12-b DACs, 10-kb flash EEPROM, and 8-b microcontroller', *IEEE Journal of Solid-State Circuits*, Vol. 33, pp.2112–2120.

Dunbar, M. (2001) 'Plug-and-play sensors in wireless networks', *IEEE Instrumentation and Measurement Magazine*, Vol. 11, pp.19–23.

Gilsinn, J. and Lee, K. (2001) 'Wireless interfaces for IEEE 1451 sensor networks', *Proceedings of the SFICON 2001*, IEEE, pp.45–50.

Gumudavelli, S., Gurkan, D., Hussain, S. and Wang, R. (2010) 'A network management approach for implementing the smart sensor plug and play', *Proceedings of the SAS 2010*, IEEE, pp.261–264.

IEEE Std. 1451.0 (2007) 'IEEE standard for a smart transducer interface for sensors and actuators common functions, communication protocols, and Transducer Electronic Data Sheet (TEDS) formats', *IEEE Instrumentation and Measurement Society Std. IEEE Std 1451.0-2007.*

IEEE Std. 21 450 (2010) 'Information technology smart transducer interface for sensors and actuators Common functions, communication protocols, and Transducer Electronic Data Sheet (TEDS) formats', *IEEE Std. ISO/IEC/IEEE 21 450.*

Kuorilehto, M., Kohvakka, M., Suhonen, J., Hamalainen, P., Hannikainen, M. and Hamalainen, T. (2007) *Ultra-Low Energy Wireless Sensor Networks in Practice: Theory, Realization and Deployment*, John Wiley & Sons, Hoboken, NJ.

Lee, K., Kim, M., Lee, S. and Lee, H. (2004) 'IEEE-1451-based smart module for in-vehicle networking systems of intelligent vehicles', *IEEE Transactions on Industrial Electronics*, Vol. 51, pp.1150–1158.

Lee, K. and Song, E. (2005) 'Object-oriented application framework for IEEE 1451.1 standard', *IEEE Transactions on Instrumentation and Measurements*, Vol. 54, pp.1527–1533.

Maxim Integrated Products (2002) *1-Wire Communication through Software*, Maxim Integrated Products Std. (2002).

Mikhaylov, K. and Tervonen, J. (2010) 'Improvement of energy consumption for over-the-air reprogramming in wireless sensor networks', *Proceedings of the ISWPC 2010*, IEEE, pp.86–92.

Mikhaylov, K. and Tervonen, J. (2011) 'Energy efficient data restoring after power-downs for wireless sensor networks nodes with energy scavenging', *Proceedings of the NTMS 2011*, IEEE, pp.1–5.

Mohammadi, S. and Jadidoleslamy, H. (2011) 'A comparison of link layer attacks on wireless sensor networks', *International Journal on Applications of Graph Theory in Wireless Ad Hoc Networks and Sensor Networks*, Vol. 3, No. 1, pp.35–56.

Motorola Semiconductor (2003) *SPI Block Guide*, Std. 03.06 (2003).

NXP Semiconductors (2007) *I^2C – Bus Specification and User Manual,* Std. rev.03 (2007).

Ovalle, D., Restrepo, D. and Montoya, A. (2010) 'Artificial intelligence for wireless sensor networks enhancement', in Chinh, H and Tan, Y. (Eds): *Smart Wireless Sensor Networks*, InTech, Rijeka, Croatia.

Potter, D. (2002) 'Smart plug and play sensors', *IEEE Instrumentation and Measurement Magazine*, Vol. 5, pp.28–30.

Ptasinski, K. and Sassi, J. (2002) '*Plug and Play I2C Slave*', Sweden Patent 6 363 437, March 26, 2002.

Rajkamal, R. and Ranjan, P. (2011) 'A framework of energy efficient wireless sensor network architecture for continuous monitoring applications', *European Journal of Scientific Research*, Vol. 61, No. 1, pp.60–67.

Ross, S., de Carvalho, A., Silva, A., Batista, E., Kitano, C., Filho, T. and Prado, T. (2009) 'Open and standardized resources for smart transducer networking', *IEEE Transactions on Instrumentation and Measurements*, Vol. 58, pp.3754–3761.

Song, E. and Lee, K. (2008a) 'Understanding IEEE 1451-networked smart transducer interface standard – what is a smart transducer?', *IEEE Instrumentation and Measurement Magazine*, Vol. 11, pp.11–17.

Song, E. and Lee, K. (2008b) 'Stws: a unified web service for IEEE 1451 smart transducers', *IEEE Transactions on Instrumentation and Measurements*, Vol. 57, pp.1749–1756.

Stepanenko, A., Lee, K., Kochan, R., Kochan, V. and Sachenko, A. (2006) 'Development of a minimal IEEE 1451.1 model for microcontroller implementation', *Proceedings of the SAS 2006*, IEEE, pp.88–93.

Texas Instruments (2009) *eZ430-RF2500 Development Tool user's guide (SLAU227E)*, Texas Instruments, Dallas, Texas, USA.

Texas Instruments (2010) *MSP430x22x2/MSP430x22x4 Datasheet (SLAS504D)*, Texas Instruments, Dallas, Texas, USA.

Texas Instruments (2011) *CC2500 Datasheet (SWRS040C)*, Texas Instruments, Dallas, Texas, USA.

Wobschall, D. (2008) 'Networked sensor monitoring using the universal IEEE 1451 standard', *IEEE Instrumentation and Measurement Magazine*, Vol. 11, pp.18–22.

Yi, Q., Kejie, L. and Tipper, D. (2007) 'A design for secure and survivable wireless sensor networks', *IEEE Transactions on Wireless Communication*, Vol. 14, No. 5, pp.30–37.

Yun, Z., Yuguang, F. and Yanchao, Z. (2008) 'Securing wireless sensor networks: a survey', *IEEE Communication Surveys and Tutorials*, Vol. 10, No. 3, pp.6–28.

Yunseop, K., Evans, R. and Iversen, W. (2008) 'Remote sensing and control of an irrigation system using a distributed wireless sensor network', *IEEE Transactions on Instrumentation and Measurements*, Vol. 57, No. 7, pp.1379–1387.

Yurish, S. (2012) *Smart Sensor Systems Integration: New Challenges*. Available online at: http://www.iaria.org/conferences2011/filesICN11/Keynote_SergeyYurish.pdf (accessed on 30 January 2012).

Appendix A

No.	*Device*	*Device description*	*ID Request*		*ID Response*	*Used device ID method*
			Address	*Request sequence*		
1	SCP1000	Pressure sensor	0x11	R_0x00NNNN	0x0300	Register contents
2	VCNL4000	Proximity + light sensor	0x13	R_0x81NN	0x11	Register contents
3	LM83	Temperature sensor	0x18-0x1A, 0x29-0x2B, 0x4C-0x4E	R_0xFENN;R_0x05NN	0x01;0x7F	Register contents
4	MCP98242	Temperature sensor	0x18-0x1F	W_0x01FFFF;R_0xNNNN	0x07FF	Inaccessible bits
5	STTS424E02	Temperature sensor	0x18-0x1F	W_0x06;R_0xNNNN; W_0x07;R_0xNNNN	0x104A;0x0000 or 0x0001	Register contents
6	LSM303DLH	Accelerometer + magnetometer	0x18,0x19, 0x1E,0x1F	R_0x0ANNNNNN	0x0A0B0C	Register contents
7, 8	LSM320 /LSM330DL	Accelerometer + gyroscope	0x18,0x19	R_0x20NNNNNN	0x070000	Register contents
9	TS3000B3A	Temperature sensor	0x18-0x1F	W_0x06;R_0xNNNN; W_0x07;R_0xNNNN	0x00B3;0x2903	Register contents
10	SE98	Temperature sensor	0x18-0x1F	W_0x06;R_0xNNNN; W_0x07;R_0xNNNN	0x1131;0xA102	Register contents
11	MAX6650	Temperature sensor	0x1B, 0x1F, 0x48,0x49	R_0x12NN;R_0x14NN	0x00;0x1F	Register contents
12	MMA8452Q	Accelerometer	0x1C,0x1D	R_0x0DNN	0x2A	Register contents
13	ADXL345	Accelerometer	0x1D,0x53	R_0x00NN;R_0x2CNN	0xE5;0x0C	Register contents
14	CMR3000	Gyroscope	0x1E,0x1F	R_0x00NNNN	0x0X21	Register contents
15, 16	HMC5843/ HMC5883	Magnetometer	0x1E,0x3D, 0x3C	R_0x10NNNNNN	0x483433	Register contents
17	PCF8575C	GPIO extender	0x20-0x27	W_0xXXXX;R_0xNNNN	0xXXXX	Register existence
18	DS3501	Temperature sensor	0x28-0x2B	R_0x00NNNNNN	0x400000	Register contents
19	APDS-9301	Light sensor	0x29,0x39, 0x49	R_0x0ANNNN	0x0500	Register contents
20	HMC6343	Compass	0x32	R_0x04NNNN	0x1101	Register contents
21	MAX17043	Fuel gauge	0x36	R_0x0CNNNN	0x971C	Register contents
22	TCM8230	CMOS camera	0x3C	R_0x00NNNNNN	0x701040	Register contents
23	BMA250	Accelerometer	0x38-0x3F	R_0x00NNNN	0x0321	Register contents
24	MAX6633	Temperature sensor	0x40-0x4F	R_0x02NNNNNNNN	0x10002800	Register contents
25	ISL29002	Light sensor	0x40-0x47	W_0xFFXX;R_0xNN	0xXX	Register existence
26	STMPE801	GPIO extender	0x41, 0x44	R_0x00NNNNNN	0x010802	Register contents
27	TMP102	Temperature sensor	0x48-0x4B	R_0x02NNNN	0x60A0	Register contents

Appendix A (continued)

No.	*Device*	*Device description*	*ID Requests*		*ID Response*	*Used device ID method*
			Address	*Request sequence*		
28	TMP100	Temperature sensor	0x48-0x4F	W_0x01;R_0xNN	0x80	Inaccessible bits
29	MAX6625	Temperature sensor	0x48-0x4B	W_0x00FF;R_0xNN	0x03	Register contents
30	MAX6642	Temperature sensor	0x48-0x4F	R_0x02NNNNNNNN	0x46007800	Register contents
31	ADT7411	Temperature sensor	0x48, 0x4A, 0x4B	R_0x23NNNN	0xC762	Register contents
32	LM75A/B	Temperature sensor	0x48-0x4F	W_0x02;R_0xNN;W_0x05;R_0xNN	0x4B;0x00	Register contents
33	SE95	Temperature sensor	0x48-0x4F	W_0xFE;R_0xNN;W_0x05;R_0xNN	0x4B;0xA1	Register contents
34	SA56004X	Temperature sensor	0x48-0x4F	R_0xFENN;R_0x05NN	0xA1;0x46	Register contents
35, 36	MCP9801/ TCN75	Temperature sensor	0x48-0x4F	W_0x02FFFF;R_0xNNNN	0xFF80	Inaccessible bits
37	STDS75	Temperature sensor	0x48-0x4F	W_0x02;R_0xNNNN; W_0x03;R_0xNNNN;	0x4800;0x5000	Register contents
38	AT30TS75	Temperature sensor	0x48-0x4F	W_0x12;R_0xNNNN; W_0x13;R_0xNNNN	0x4B00;0x5000	Register contents
39	24XX256	EEPROM	0x50-0x57	W_0x001FXX;R_0x001FNN	0xXX	Register existence
40	DS1077	Oscillator	0x58-0x5F	R_0x02NNNN	0x1800	Register contents
41	Si1141	Proximity sensor	0x5A	R_0x01NNNN	0x4101	Register contents
42	Si1142	Proximity sensor	0x5A	R_0x01NNNN	0x4102	Register contents
43	Si1143	Proximity sensor	0x5A	R_0x01NNNN	0x4103	Register contents
44	MPR121	Touch sensor	0x5A-0x5D	R_0x5CNNNN	0x1004	Register contents
45	MPL115A2	Barometer	0x60	R_0x0CNXXXNXXX	0x0NNN0NNN	Inaccessible bits
46	MCP4725	DAC & Memory	0x60-0x67	R_0xNNNNNNNNNN	0x80NNNN0800	Register contents
47	IMU-3000	Motion sensor	0x68,0x69	R_0x00NN	0x34	Register contents
48	BMP085	Temperature + pressure sensor	0x77	R_0xAANNNN	0x20E3	Register contents

Note: Used designations: 0xXX – some specified data (hex); 0xNN-any data (actual value not important); R_0x01NN – issue together with address read strobe, after that send byte “0x01” and receive 1 data byte; W_0x07XX issue together with address write strobe, after that send byte “0x07” and 1 data byte with value “0xXX”.