DESIGN AND IMPLEMENTATION OF THE PLUG&PLAY ENABLED FLEXIBLE MODULAR WIRELESS SENSOR AND ACTUATOR NETWORK PLATFORM
Konstantin Mikhaylov and Juha Petäjäjärvi

Abstract:
In this paper we address the problem of increasing the flexibility of the contemporary wireless sensor and actuator networks (WSANs) in regard to the design of the nodes. For this we propose the concept of a Plug&Play enabled modular WSAN node platform. According to our concept, the new WSAN nodes with desired functionalities can be built by stacking together the different hardware modules encapsulating power sources, processing units, wired and wireless transceivers, sensors and actuators, or even sets of these. Once a node is built, it automatically discovers and identifies all the connected hardware modules, obtains required software and tunes its own operation taking into account the node’s structure, available resources and active applications. In this paper we first present the concept and then report the developed hardware and software architectures and the most critical mechanisms enabling implementation. Also we discuss the practical implementations and report the evaluation results for the prototyped solution. Our results show that the developed platform is much more feature and resource rich than the existing ones, which is achieved at a cost of increased consumption and size. We believe that the unique features of the proposed platform have the potential to drastically change the procedure of WSAN development, especially when it comes to experimenting and developing dynamic WSANs with a heterogeneous structure. In this respect the hardware identification and reconfiguration capabilities conceived in the platform can be utilized in full and may drastically increase the performance of WSANs, if combined with novel control and optimization schemes yet to be developed.

I. Introduction
The distributed measurement and control systems are nowadays on the eve of dramatic changes. Departing from the traditional wireless sensor networks (WSNs) of the early 2000s [1] which were predominantly single-purpose open-loop information collection systems, these systems have evolved into the wireless sensor and actuator networks (WSAN) featuring very wide range of functionalities. In contrast to the WSNs, the contemporary WSANs are often designed as closed loop or hybrid open-loop/closed-loop systems and feature heterogeneous network structure with the nodes differing in terms of their hardware (HW) and software (SW), the available resources, and their tasks. Also the communication landscape of WSANs has changed dramatically over the recent years. The development of the new wireless standards and proprietary radio technologies as well as the initiatives for amalgamating the communication mechanisms (e.g., 6LoWPAN [2]) and standardizing the data representation (e.g., IEEE 1451[3] or OGC Sensor Web enablement [4]) emphasize the importance and enable cooperation between the co-located WSANs. In future, this may result in merging of individual WSANs and formation of a single communication and data landscape which will bring to life the long-talked-about Internet of Things (IoT). These two dramatic transformations inevitably make the WSANs of the future more complex and emphasize the importance of looking on them from the control perspective in order to come up with more advanced control and optimization mechanisms. Also this calls for updating the long accepted procedures for WSAN design and development.

When it comes to the design, the contemporary WSANs are subject to the two opposite trends. On one hand, many real-life applications require hundreds or even thousands of nodes being installed throughout the area of interest to enable reliable detection and classification of observed phenomena. For the success of these applications, high efficiency and low cost of manufacturing, deploying and maintenance are critical. To enable this, the engineers today often rely on point solutions which are built using specialized HW, custom SW and
proprietary communication protocols optimized for the particular use case scenario. This results in limited interoperability, low flexibility and poor extensibility of such designs. On the other hand, the nodes featuring generic albeit customizable architecture, communication and data encoding mechanisms, might be somewhat less efficient, but enable easy extension, modification, and seamless integration of different systems and applications. This becomes especially important for multidisciplinary and multipurpose applications, which often require having nodes of the different architectures in order to monitor a sheer diversity of physical phenomena [5].

In this study, we attempt to reconcile both trends by proposing a flexible and reconfigurable platform for WSAN nodes based on new node design approach. The specifics and the main contributions of this paper are 1) the suggested concept of the modular WSAN node featuring plug-and-play (P&P) connection of the modules, 2) the HW and SW architectures and the developed mechanisms enabling this concept, 3) the implementation of the platform prototypes and respective evaluation results. Importantly, the proposed solution enables easy migration between the dynamic modular and the static architectures by utilizing the proposed “virtual” modules mechanism. This opens the way for the further optimization and makes the proposed solution complement rather than oppose the traditional WSAN architectures.

The paper is organized as follows. Section II overviews the related works. Section III describes our vision and presents the concept. Section IV focuses on the HW architecture of the platform, discusses the faced trade-offs and introduces the most critical mechanisms developed. Section V presents the special SW architecture to be used with the flexible HW. Section VI provides details of our platform's implementation and Section VII presents results of evaluation. Finally, Section VIII concludes the paper and points out some possible directions of the further work.

II. Related works
Based on their HW architecture the WSAN platforms today can be divided into the three major groups, namely: static, semi-dynamic and dynamic. For the former ones the HW structure is pre-defined and is not intended to be changed. Good examples of such systems are e.g. Shimmer3 [6] or LoRaMote [7] nodes. The platforms which can be attributed to the second group are probably the most multiducious and are widely used for the practical research in context of WSNs and WSANs. These devices typically feature a static set of the core HW components (e.g., a microcontroller, a radio, a battery connector, optionally – few basic sensors) and enable connection of the application-specific HW (e.g., sensors or radios) via some sort of daughter card connector, by wires or by soldering the new components on the printed circuitry board (PCB). Among these platforms can be listed e.g. the well-known MICA/MICA2 [8] or TELOS[B] [9] nodes. Finally, the dynamic WSAN platforms provide the ultimate level of HW configurability and enable changing all the peripherals including, in some cases, the processing system. Further in this section we will focus specifically on these platforms. In respect to the static and semi-dynamic platforms the reader can check e.g. [10].

To enable dynamic reconfiguration of the WSAN node's HW structure, the platforms presented in [11-25] are assembled out of the functional modules, number of which ranges between two in [24] and seven in [21]). The different modules featuring specific functionalities (e.g., temperature sensing, energy harvesting or additional memory) are attached to the core module which hosts the main processor and, optionally, a radio and a primary power source. By modifying the set of attached modules, functionality of a WSAN node can be adjusted to fit particular application or use case. The features of different state-of-the-art (SotA) modular platforms are summarized in Table 1 and below we will briefly discuss few of them.

Probably the most feature-rich modular WSAN platform today is Tyndall [11,13,27]. The node is composed of the 25mm x 25mm (or 10mm x 10mm) module boards which are stacked on top of a main board hosting a microcontroller and a radio. As reported in [13], the number of different modules available for Tyndall exceeds 30 of which more than 20 are the different sensor options. Although the authors claim that the platform enables P&P module connection, no details about the mechanism are provided.
Somewhat different approach was taken by the authors of [14] while developing MASS platform. Unlike other platforms, which have a single central processing core, Edmonds et al. propose to decouple system management from in-system data fusion and signal processing by using distributed multiprocessor HW architecture. This approach permits a WSAN node to have multiple independent processing units and enables the modules to perform their tasks in parallel and request resources from other modules when necessary. Communication between the modules is implemented via Inter-Integrated Circuit (I2C) bus [14], which, as reported e.g. in [28], has quite limited module-identification and extension capabilities.

The major feature of the platform developed by R. Martinez-Catalá and J. Barton and reported in [19] is its design. The platform consists of six 20 x 20 mm module boards located on the six faces of a cube. The batteries providing power to the node are stacked in a plastic frame placed in the center of the cube. The modules are interconnected by the means of connection tracks located on the sides of each board.

The specifics of Cookie platform [20] is the architecture of its base module. In order to improve computing flexibility, the main processing module hosts both an 8051 microcontroller and a field-programmable gate array (FPGA). The latter handles all processing related to digital sensors, while the microcontroller manages node’s communication and gets data from the analog sensors.

To improve energy efficiency and autonomy of modules, the developers of CoSeN architecture [23] placed on each module a Module Management Controller (MMC). The MMC manages local tasks of its board and identifies it to the system. Similarly to MASS platform, for communication between an MMC and the main processors CoSeN relies on the I2C bus.

Although the solutions proposed in [26] and [30] cannot be considered as a full-featured modular WSAN platforms, the approach should be noted. In [30] the authors reported the design of the IEEE 1451 [3] based smart transducer interface module (STIM) enabling P&P connection of the various transducers. Albeit the authors neither enabled wireless communication nor developed a full featured sensor network platform, the approach is worth noting. Furthermore, in [26] the authors proposed extending the IEEE 1451 smart transducer interface standard to power sources domain. As demonstrated by the authors in [26], this enables P&P connection of the various power supply modules to a WSAN node.

A good example of a commercial solution following the same philosophy which gets occasionally used also for the research purposes is Arduino [29].
The proposed concept is depicted in Fig. 1. The node can also feature the possibility of reporting its structure to the other nodes and network sink, discover the structures of the neighboring nodes, and support SW application downloading from WSAN. The key components of the proposed concept are depicted in Fig. 2.

Note, that when compared to the SotA platforms discussed in Section II, the proposed concept has two critical differences. First of all, the proposed concept does not focus exclusively on the HW modularity, but brings together HW and SW modularity to enable full-featured P&P module connectivity for WSANs. The other critical difference is that the proposed WSAN nodes neither require to possess any information on their own HW structure nor need to have all the SW drivers and applications hardcoded. Instead, the proposed
concept enables the SW to be obtained and configured dynamically after the node has identified its structure and taking into account the HW architecture of the node, its available resources and the potential needs of the network and applications. Such flexibility inevitably introduces additional requirements both for HW and SW components of the system and requires more complex and advanced mechanisms to be involved. In the following sections we introduce and describe our proposed solutions.

Figure 1. High level state diagram of proposed WSAN node's operation.

Figure 2. Proposed modular WSAN node concept.
IV. Proposed hardware architecture and mechanisms

As discussed in Section II, in the literature there are two fundamentally different approaches towards WSAN HW architecture. The dominating one is based on hierarchical system in which a single main processing unit (MPU) controls operation of the whole node. The other option (used in MASS) is to have a distributed decentralized multiprocessor system with multiple independent processing units. Since the latter inevitably increases the cost, energy consumption and complicates task management, for our system we have selected the former option. Nonetheless, further we will show that the proposed mechanisms can be used for a distributed architecture as well.

The other critical problem in the context of the proposed concept is the need for a mechanism enabling HW modules discovery, identification and control. The attempt to develop a non-invasive solution for this problem was reported by us in [28]. Even though the developed mechanisms enabled MPU getting some information about the digital peripherals around it, the data was not very reliable and the solution scaled up poorly. In their turn, the discussed SotA WSAN platforms have limited means for HW module identification based either on I2C bus or on IEEE 1451.

In both cases, a unique address is used for module identification. The first problem related to this approach is the need for a mechanism of unique address generation and address collision resolution in case if two or more modules happen to get the same address. The other issue is the limited address space and absence of an efficient device discovery mechanism for I2C bus. Due to these reasons, for our architecture we have chosen a fundamentally different approach which is based on the natural “identifier” of the module – its position relative to the base board and the other modules.

The final open question is how one should handle the communication between the peripherals on the modules and the MPU. The trivial solution is to relay all the control over the peripherals on-module microcontrollers, which are controlled by MPU by means of identification bus. This solution makes inter-module communication a real bottleneck for multi-module nodes thus compromising the scalability. The other option is to enable multiple direct communication interfaces between MPU and peripherals which can function in parallel. The major problems in this case are implementation of interfaces sharing between the peripherals and mapping of the interface to a specific peripheral by MPU. For our solution we have taken the second path as discussed further.

The simplified structural diagram of the proposed architecture is presented in Fig. 3. A WSAN node consists of a base module built around the MPU and a set of peripheral modules which are stacked on top (or on bottom) of the base module. Each peripheral module includes a module control and identification unit (MCIU), which enables its identification and also provides the MPU with basic means for controlling operation and consumption of a peripheral modules (e.g., enables switching on/off power supply of a specific peripheral). Physically, the MCIU can be implemented on a microcontroller, programmable logic device or an integrated circuit. Besides the MCIU a peripheral module contains a set of peripherals such as wireless transceivers, sensors, actuators, power sources, memory or application specific processors. Note, that unlike the existing modular platforms which have function-specific modules, modules for our platform can host peripherals of different types (e.g., a radio, a sensor and a power source may reside on the same module).
The communication between MPU and peripheral modules is implemented via the specially developed Intelligent Modular Periphery (IMP) Interface (IMPI). Functionally, the lines of IMPI can be divided into four major groups. First come the power supply lines, which are used for a) getting power from power source modules ($V_{in}$ line) to the main board, and b) powering the connected peripheral modules ($V_{out}$ line).

The second set of lines is reserved for IMP-bus, which a) is responsible for detecting the connected modules and their connection order and b) enables module and peripheral identification and control over them. The IMP-bus is designed as a daisy chain interface based on the Serial Peripheral Interface (SPI) hardware bus. This was done due to two reasons. First, the communication between MPU and MCIU needs to be fast, error-free, energy efficient and implementable for the various existing microcontrollers acting as MPU. As shown in [32], in this respect SPI is the most efficient of all standard interfaces. Second, daisy chain connection ensures that the data are transferred sequentially between the MCIUs which is used to determine their connection sequence and relative position for addressing. The procedure for module discovery and the state diagram of the MCIU are illustrated in Figs. 4 and 5, respectively.

Once the MPU has detected the number of connected modules and their connection order, it proceeds with identification of peripherals available on each module and detection of their connection interfaces. For this, MPU goes through MCIUs of each connected module one after another, selecting them as illustrated in Fig. 4, and reads the number of peripherals available. Then the MPU obtains periphery description data (PDD) pre-stored in the non-volatile memory of MCIU for each peripheral device. The PDD is composed of the two major blocks, namely peripheral connection descriptor (PCD) and peripheral service data (PSD). The former one lists all communication interfaces used by a peripheral. The PSD provides additional information about the peripheral (e.g., name, identifier, manufacturer, description, calibration data, and may include SW drivers for working with the peripheral). The format of the MCIU registers used for module identification and control is presented in Fig. 6. In order to reduce energy consumption, MCIUs stay in the low-power sleep mode most of the time and leave it only when MPU activates IMP bus by pulling select (SEL) line to ground. The structure of the PDD and the command set for MCIU are presented in Figs. 7 and 8, respectively. Note, that we do not restrict in any way the formatting of the PSD; e.g. for this purpose one can easily use IEEE 1451 or OGC SensorML.
Figure 4. Illustration of the module discovery, data exchange over IMP bus and IMP packet format.

Figure 5. State diagram of an MCIU.

Figure 6. Registers of a module control and identification unit.
The lines in the third set are intended for enabling direct communication between the MPU and the peripherals. These lines feature different data transfer interfaces grouped by interface type. Prior to starting the communication with the peripherals, MPU needs to map the data interfaces to particular peripherals using the data from PCD. This is done based on the information on the interfaces used by each particular peripheral and the respective commutation data showing whether particular interface goes to the next
module or not (e.g. the first peripheral module depicted in Fig. 3 reserves 2 GPIO lines and UART, and forwards all other lines further). Since the connection order of the modules in chain and order of the peripherals on a module are now both known to MPU, it can define which data interface lines should be used for communicating with each peripheral.

The final group of lines consists of one or several interrupt (IRQ) lines which are used by the peripherals to draw the attention of the MPU and inform it about important events.

Note that the proposed interface can be adapted to multi-MPU architectures with minimum efforts. For this, each MPU detecting the IMP SEL line pulled down by another MPU should switch to bypass state and behave as an MCIU (see Fig. 5).

V. Proposed software architecture

Dynamism of the proposed HW architecture has one critical consequence when this comes to the design of the SW [33]. Namely it prevents use of the traditional WSAN SW development practices which are based on two common assumptions, i.e. the knowledge of the node’s HW structure and the immutability of this structure throughout the node’s operation. In order to address this issue below we propose our solution for this problem.

Figure 9. Proposed software architecture and interfaces between its components

The proposed SW architecture is depicted in Fig. 9. The core component of the middleware is the Resource manager composed of the three major units. The first one is the Module manager, which is a low-level entity responsible for identification of the peripherals and modules (i.e., obtaining PDD), controlling power supply of the peripherals and of the whole node (e.g., dynamic voltage-frequency scaling), interpretation and prioritization of the interrupts coming from various peripherals. The second component is the Communication manager which handles all the communication of a node with external world. Based on the
information on the available peripherals, the manager decides which communication technology and which parameters to set. Depending on the resources available for the manager and on the range of modules and applications supported, the communication parameters may be assigned either statically or can be defined dynamically for each transmitted frame. The other functionalities which must be supported by Communication manager include: discovery of devices and networks, translation of addresses and routing between the different communication technologies, prevention of interference between the interfaces and handling of communication interface change if the current one is ineffective. Finally, the Application manager decides which applications and services can be launched and supervises their operation. Depending on the presence of control entities in the network, the applications executed by each node may be either defined by a node itself or dictated from remote. The former option requires a node to have a decision unit which comes up with the list of applications to execute based on the formulated set of rules. Also, in order to enable transferring the tasks between the nodes, the applications should be written in HW independent manner and handled by a virtual machine or some sort of interpreter.

Another important feature of the proposed architecture is the design of low-level communication interface drivers enabling the peripheral drivers to communicate with peripherals. The drivers for each interface (e.g. general purpose inputs/outputs (GPIOs), SPI, I2C, universal asynchronous receiver/transmitter (UART), etc.) are implemented by specific SW instances, which are accessed by the peripheral drivers through the standardized access interfaces. The identifiers of the interfaces for each particular peripheral connected to a node are defined as a part of peripheral identification procedure and are reported to the drivers by the module manager. Later, when a driver desires to communicate with a peripheral, it calls the respective low-level interface driver and provides the identifier of the interface to use. On one hand, this enables to abstract from the specifics of HW interface implementation for particular MPU and leave aside the details of peripheral’s connection when implementing the peripheral drivers. On the other hand, this provides a mechanism for sharing the communication interfaces between the peripherals and interface access prioritization.

The peripheral drivers should be implemented as independent threads which are interfaced to the respective communication interface drivers and applications. In addition, the drivers of wired and wireless transceivers should be registered at the communication manager. If desired, the drivers and the protocol stacks may be implemented as scripts executed by an interpreter. One of the principal questions is where and how the peripheral drivers are stored. The possible options include: a remote network location, the internal memory of MPU, and memory of peripheral modules. In the former case, for downloading the drivers one can use the well-established over-the-air WSN/WSAN programming solutions [34-36].

VI. Implementation

For evaluating the proposed concept and developed technical solutions, a prototype has been designed and implemented in HW. The major characteristics of the designed prototype are summarized in Table 2.

Table 2. Characteristics of the implemented prototype system.

<table>
<thead>
<tr>
<th>MPU:</th>
<th>STM32F207/STM32F217: ARM-based 32-bit microcontroller with 1 MB Flash, 128 kB RAM, and up to 120 MHz clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main board components:</td>
<td>voltage regulator and power switches, FRAM, SRAM (optional), 1-wire controller</td>
</tr>
<tr>
<td>Main board connectors:</td>
<td>miniUSB, battery, UART (debug), 2 x IMPI (30 pins each)</td>
</tr>
<tr>
<td>Main board dimensions:</td>
<td>70x70 mm (2 layer) or 50x70 mm (4 layer)</td>
</tr>
<tr>
<td>MCIU:</td>
<td>MSP430G2453: 16-bit microcontroller with 8kB Flash, 512 B RAM</td>
</tr>
<tr>
<td>Peripheral modules implemented:</td>
<td>19 in total including: radios (IEEE 802.15.4 2450 DSSS, IEEE 802.15.4a, BLE, LoRa, WiFi, sub-GHz AFS/FSK, LTE), wake-up radio (WUR), sensors (buttons, climate, ECG, air quality), actuators (LED, screen), power (battery, PC interface, energy harvesting), ADC and SD-card</td>
</tr>
</tbody>
</table>
Figure 10. Implemented main board a) Structural diagram b) Photo of the manufactured 2 layer board.

The structural diagram and the manufactured prototype of the main board are depicted in Figs. 10 a) and b), respectively. Besides MPU, the main board contains a unique dynamic voltage regulation circuitry enabling the MPU to control the supply voltage provided to itself and the peripheral modules. Also the board hosts a ferroelectric random-access memory (RAM) and, optionally, a static RAM. Note, that all the peripherals
installed on the main board are treated by the MPU SW as "virtual" modules (i.e., PDD for them are pre-stored in non-volatile memory of MPU). This enables to handle these peripherals in SW as if they were the external HW modules connected via IMPI. As one can easily see, the very same solution can be utilized on the nodes with the static architectures in order to enable them running the SW developed for the modular nodes without any changes. The boards have two IMPI connectors placed on the opposite faces of PCB thus enabling stacking the modules both upwards and downwards from the main board. Each IMPI connector features the power rails, IMP bus, interrupt line, eight GPIO lines, one SPI, one I2C bus, one UART, and one 1-wire and one secure digital input output interface shared between two connectors. The two versions of boards have been developed - a two-layer test board enabling easy access to all the signals, and a more compact 4-layer board.

Besides the main board we have designed and built in the HW 19 example peripheral modules. They include: IEEE 802.15.4 2450 DSSS (ZigBee), IEEE 802.15.4a ultra wide band (UWB), Bluetooth Low Energy (BLE)/Smart, LoRa low-power wide area network (LPWAN), IEEE 802.11 b/g/n, 433 and 868 MHz amplitude and frequency shift keying transceivers; light, temperature, pressure, humidity, air quality and electrocardiogram/oximetry sensors; AAA battery power; LEDs; miniSD card holder; and USB-UART interface to personal computer. All peripheral modules have MSP430G2453 as MCIU. The firmware implementing IMPI communication and peripherals’ PDDs are stored in MCIU’s internal electrically erasable programmable read-only memory (EEPROM). The designed modules are shown in Fig. 11. Note, that unlike the SotA solutions, the developed platform supports connection of multiple identical peripheral modules. The only factors limiting the scalability of the node are the number of data interface lines taken by peripherals and the total power consumed. Aside of the MCIU and the peripherals, each designed peripheral module includes a special circuitry, enabling it to detect whether this module is the top one in the stack and commutating the data out line of the MCIU either to the data in line of the MPU for the top board or to the data in line of the next MCIU in the stack otherwise. The very same mechanism is employed in the case of IMP chain break in order to insulate the modules following the chain break.

Figure 11. Modules for the designed platform (left), few examples of assembled nodes and the laptop with the graphical user interface of the developed demo application (right)\(^1\).

The firmware implementing the most critical components of the proposed SW architecture was developed. The code is written in C language and operates on top of FreeRTOS embedded operation system (OS). The developed SW includes the low level drivers and respective abstractions for IMP, SPI, I2C, UART interfaces

\(^1\) The live demonstration was first presented at IPSN 2015 conference [37]
and GPIOs, the peripheral drivers for designed modules, module, application and communication managers and few demo applications. Albeit for the current version of the code the SW is still majorly statically placed inside the MPU internal program memory, the possibility of obtaining the pre-compiled peripheral drivers from the memory of the MCIUs as well as for the wireless transfer of the SW between the nodes (based on the mechanism reported in [36]) have been successfully tested for the developed platform.

VII. Evaluation

Unfortunately, the lack of the proper evaluation mechanisms and procedures does not enable us to assess quantitatively the level of flexibility enabled by the proposed platform. Also, since the respective information is typically absent from the literature, we cannot make a fair comparison with the SotA platform in this respect. Therefore, we focused on evaluating the most important quantitative performance metrics in the context of WSAN, namely the performance, available resources and energy efficiency.

Figs. 12 a), b) and c) graphically compare the characteristics of the MPU used in our platform and the ones operating in the SotA modular and well-known semi-dynamic WSAN platforms. The comparison is based on the data obtained from the data sheets of the microcontrollers. One can clearly see that in terms of available processing power, our platform significantly exceeds most of the traditional platforms. The only platform with comparable performance is the SunSpot node, which hosted a high-end processor for running the Java applications. The peak processing performance of 150 Dhrystone million instructions per second (DMIPS) (as reported in [38]) enable using our platform even for computation-hungry applications like live video or audio processing. Also the amount of on-chip memory for our platform is at least 8 times higher than what is available for the traditional platform. Nonetheless, there are three major costs of this, namely the increased sleep mode consumption, higher node’s cost and bigger linear dimensions of the microcontroller and the board.

Since comparison of the performance characteristics based only on the information obtained from the data sheets does not provide a comprehensive picture, we evaluated the practical performance of the core processor of our solution using the CoreMark[39] benchmark. For this, we adapted the code from the STM3220F evaluation board (Keil UVision toolchain) port[40] of the benchmark. The code was compiled with GCC compiler version 4.8.3 with the following flags: -O3. While running with 120 MHz clock and having the code in Flash, the benchmark showed the controller’s performance of 26 CoreMark points (i.e., iterations per second). The time of the experiment was measured using the timer clocked from the main clock and was double-checked with an oscilloscope. Albeit the obtained results are somewhat lower than the ones reported by the manufacturer in [38], they match well with the results of the tests obtained for the STM3220F evaluation board (refer to the comments in core_portme.h file of the respective port). Even though, comparison of the obtained results (available from [39]) with the processors used in the traditional WSN nodes shows that in terms of the processing power our platform significantly outperforms them.
Figure 12. Comparison of MPU with the controllers in SotA WSAN platforms a) maximum clock and processor resolution, b) RAM and Flash volume, c) minimum supply voltage and low power mode current.
In order to understand deeper the effect of the supply voltage on the computing performance for the implemented platform we have conducted a set of measurements. The main board of the platform with the MPU operating in different modes with different core voltages provided by the voltage regulator was connected to the N6705B power analyzer and the consumed current was measured. The respective results for MPU in active mode and in two low power modes are presented in Figs. 14 a) and b), respectively. First of all, one can see that the designed platform can operate over wide range of input voltages. Second, the presented results reveal that the consumption in active mode strongly depends on the relation between the level of the voltage provided by power source and the core voltage requested by the microcontroller. The consumption is maximal when the voltage regulator has to convert the voltage up. This needs to be accounted for when designing the algorithms implementing the control over voltage. Also the measurements show that in sleep mode the minimum consumption of the board is below 50 µA. The further experiments have shown that approximately 95% of this current is consumed by the voltage regulator. This can be treated as the cost for the flexibility of the voltage control enabled. Note that the major purpose of the voltage regulator is to provide the voltage required by the peripheral devices, which often have much more narrow supply voltage range than the MPU. Thus the addition of the regulator is well reasoned. On the other hand, even having this consumption, a node can operate more than 3 years when powered from AA batteries or used with energy harvesting. Therefore, considering all pros and cons, we find this level of consumption to be admissible. Nonetheless, in future we will consider adding a mechanism enabling temporal deactivation of the power regulator when the node is in sleep. Also we have measured the consumption of the MCIU installed on the modules to enable their identification and control. When IMP bus is inactive, which is its normal state, a single MCIU consumes a mere 300 nA. Thus even attachment of multiple modules will not affect significantly the total energy consumption and the life time.

At the next stage of the evaluation we implemented three illustrative applications and measured both the consumption and the time of the code’s execution. The first test application represented one of the most common embedded test applications (e.g., compare with the “Blink” application for TinyOS) and involved a periodic control of an LED. With a period of one second a red LED (with 820 Ohm resistor in series) was periodically switched on and off. The second application demonstrates one of the most common WSN use cases which is the periodic sensing. For this application the node was equipped with the IEEE 802.15.4 radio
transceiver and the environment sensor modules. With the period of three seconds the microcontroller wakes up, reads the measurements from humidity, pressure, temperature and light sensors and broadcasts them wirelessly. Finally, the third application illustrated the use of the node as a localizable tag. For this, an IEEE 802.15.4a compatible UWB radio transceiver module has been attached to the node. With the period of three seconds the node broadcasted a radio packet which were received by one or several receivers and can enable node’s localization (refer, e.g., to [41]). In all three applications the time was tracked using the real-time clock (RTC) available in the used microcontroller, which was clocked from external 32 kHz oscillator. Also for all the applications in between the wake-up events the microcontroller stayed in Stop\(^2\) mode. After wake up the microcontroller was clocked from the internal HCI oscillator. The consumption profiles have been measured with the Agilent N6705B power analyzer and are illustrated in Figs. 15 a), b) and c).

\(^{12}\) Albeit the Stop mode does not give the lowest consumption possible, this is the only mode where registers (including the GPIO control) retain their states. Due to this reason, for our tests we have used this mode.
Figure 15. Current consumption profiles for the three illustrative applications (2.8 V core voltage, voltage regulator bypassed): a) periodic LED switching (red LED with 820 Ohm series resistor, 2 second period with 50% duty cycle), b) periodic reading of the sensor data (from HTU21 humidity (11 bit, interfaced over 400 kHz I2C), MPL115 pressure/temperature (interfaced over 8 MHz SPI) and BH1715 light sensor (low resolution, interfaced over 400 kHz I2C)) and their broadcasting with IEEE 802.15.4 transceiver (AT86RF233, interfaced over 8 MHz SPI, transmit power 3 dBm, 8 byte link layer payload, radio in TRX_OFF between transmissions), c) periodic broadcasting of a IEEE 802.15.4a UWB packet (DWM1000 radio module, interfaced over 15 MHz SPI, radio configuration: data rate 6.8 Mb/s, PRF=64 MHz, channel 3, 128 symbol preamble, SLEEP in between TX, power to match -41 dBm in 1 MHz bandwidth, 64 byte link layer payload).
The presented results clearly show that the time taken by the actual data processing for either of the three illustrative applications is very small. Meanwhile, communication with the peripherals (i.e., sensors and radio transceivers) takes very significant time. The two major reasons for this are the low clock rate of the interfaces (especially for the I2C interfaces, which is limited by 400 kHz) and the design of the interface SW drivers in the used release, which are based on extensive use of the memory allocation functions. Based on the presented results of the measurements was conducted the brief analysis of the possible lifetime of the system when powered from the batteries. For this, we have used the linear consumption model (equation (8) in [42]), the average power consumption measured and the data about the capacities of the different batteries obtained from ([43],[44] and [45]). Albeit this method may not be precisely accurate since it does not account for such effects as the current consumption variations due to the voltage reduction, battery ageing and recovery effects (refer e.g., to [42],[43]), the obtained results can be used as the first-order approximation. The results of analysis are summarized in Table 3.

Table 3. Estimated life times for the three illustrative applications.

<table>
<thead>
<tr>
<th>Application</th>
<th>Average application power consumption, mW</th>
<th>Lifetime if powered from 2 AG13 button Alkaline batteries1</th>
<th>Lifetime if powered from single CR2032 Lithium battery2</th>
<th>Lifetime if powered from 2 AAA Alkaline batteries3</th>
<th>Lifetime if powered from 2 AA batteries4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Periodic LED switch</td>
<td>2,3464</td>
<td>140 hours</td>
<td>238 hours</td>
<td>71 days</td>
<td>115 days</td>
</tr>
<tr>
<td>Environment sensing and wireless reporting every 3 seconds</td>
<td>7,448</td>
<td>44 hours</td>
<td>75 hours</td>
<td>22 days</td>
<td>36 days</td>
</tr>
<tr>
<td>UWB localization tag reporting every 3 seconds</td>
<td>5,32</td>
<td>62 hours</td>
<td>105 hours</td>
<td>31 days</td>
<td>50 days</td>
</tr>
</tbody>
</table>

1 – stored power 165 mWh per battery [43]
2 – stored power 2.8 V x 200 mAh = 560 mWh [44]
3 – stored power 2 Wh per battery [45]
4 – stored power 1.2 V x 2.7 Ah = 3.24 Wh per battery [42,45]

Comparison of the obtained lifetime estimations with the ones reported in the literature (e.g., [46-49]) for the traditional general-purpose WSN platforms appear to be rather complimentary for our solution. E.g., in [46] the lifetime of a Mica2 node equipped with two AA batteries and executing the TinyOS “Blink” application is estimated to be below 26 days and only 10 days for the “SenseToRfm” application (involves periodic broadcasting of the light sensor data). The similar results are reported in [49]. Nonetheless, comparison of our system with the WSN platforms optimized for low power consumption (e.g., TUTWSN, having the lifetime of over half a year for 1 second periodic reporting of the temperature [50]) calls for the further reduction of the power consumption both in respect to the SW and HW designs. First of all, we have to note that by no means the given illustrative consumption values for our platform can be treated as the minimal ones. E.g., the modifications of the SW to take advantage of using DMA and putting the main core to sleep while the communication with the sensors and the radio transceivers is ongoing can substantially reduce the power consumption in active mode. The modifications of the HW to enable the GPIOs retain their states (e.g., using flip-flops) even when the microcontroller’s pins are released can enable putting the core in Standby mode with lower consumption. An alternative for this is to use for the power-critical applications another more optimized MPU, whilst the current design will be used for the power processing power demanding applications. For this reason, we have initiated the development of the low power version of the main board having the CC2650 system on chip (SoC) from Texas Instruments as the MPU. The SoC features more efficient power control both for active and sleep modes on one hand, and enables to omit the communication with the radio transceiver over external interfaces which is shown to take substantial time (see Figs. 15 b) and c)). The new main board is based on the same principles and uses the same connectors as the STM32 main board (but, since the SoC has less GPIOs, the new board can accept the modules only from top) which enables the P&P connection and use of all the already developed modules with it.

Finally, we used the N6705B for monitoring the consumption of the platform with few different radio transceiver modules during radio communication. The respective results are presented in Figs. 16 a) – e). Note that the waveforms presented in Figs. 16 a), c)-e) illustrate the consumption of only the MPU and the peripheral module, omitting the power regulator. The reason for this can be seen from Fig. 16 b) which illustrates the consumption of a BLE-enabled node including the power regulator. One can clearly see that
constant switching of regulator makes the waveform hard to comprehend. Meanwhile, the difference of the total consumption with and without the regulator does not differ for more than few percents.

![Figure 16. Current consumption of the node with different radio modules](image)

**VIII. Conclusions**

In this paper we have presented the concept, proposed the hardware and software architectures, and discussed implementation of the new modular WSAN node platform. The developed system enables fast and efficient construction of the WSAN nodes with desired characteristics out of the hardware modules featuring the various power supply sources, processing units, wireless transceivers, sensors and actuators, or other devices. All the modules are automatically discovered, identified and taken in use by the main controller. The software modules can be automatically obtained and the operation of the node can be optimized accounting for the available hardware and resources, and the needs of the network and applications.

If compared to the state-of-the-art systems and platforms, the one presented in this paper provides much more flexibility and makes the WSAN node fully aware of its capabilities and limitations. Also, unlike the existing solution (e.g., based on the IEEE 1451 [30]), the proposed one can be applied to the whole diversity of the potential WSAN peripherals. Importantly, the developed solution enables easy migration between the dynamic modular and the static architectures by utilizing the proposed “virtual” modules mechanism. This

---

3 The peripheral modules used in the measurements are based on AT86RF233 (IEEE 802.15.4 2450 DSSS), DWM1000 (IEEE 802.15.4a UWB), RN2483 (LoRa), RN4020 (BLE) transceivers. During all the measurements the MPU was operating in active state at 120 MHz clock. For Figs. 14 a), c)-e) the measurements were done for 3.55V core and peripheral supply voltage. For Fig. 14 b) the regulator’s input voltage was 5 V and the regulator was converting it to 3.55 V. For all the cases the transmitted packets carried 10 bytes payload. The following settings were used for the different technologies:
- IEEE 802.15.4 2450 DSSS: transmit power $P_{tx}=3$ dBm
- IEEE 802.15.4a UWB: transmit power $P_{tx}=-41.3$ dBm/MHz, 1024 symbol preamble, 16 MHz PRF, 850 kbit/s rate
- BLE: $P_{tx}=7$ dBm
- LoRa: $P_{tx}=14$ dBm, spreading factor 7, bandwidth 125 kHz, 4/5
opens the way for the further optimization and makes the proposed solution complement rather than oppose the traditional WSAN architectures. In our opinion, the use of the proposed solution is especially beneficial for enabling experimenting with the dynamic WSANs with the heterogeneous structure, which has been the major purpose of the platform development. The presented evaluation results also show that the platform has much more processing power and memory than the traditional WSAN solutions. This enables loading sufficiently complex control and data processing algorithms on the nodes. The advanced voltage control system of the nodes enables to dynamically tune the level of voltage supplied to the main controller and the peripherals. The major costs for this flexibility are the higher sleep mode consumption, bigger size and production cost. Even though, the presented analytic results show that the platform can support months-long operation when powered from batteries. So far we have already implemented 19 different peripheral modules, which can be connected to the main board in any combination. The implemented modules include 9 wireless transceiver options for state-of-the-art and perspective communication technologies, 4 sensor modules, 3 power supply and interface options, 2 actuators, ADC and memory card modules.

Although much has been already done, there is still a long way ahead. So far we have primarily focused on enabling the mechanisms which can provide the desired level of flexibility for hardware and software on the node level. The next and the most challenging step is to take the advantage of the developed solutions at system and network level. Indeed, the developed platform has two distinguishing features. The first one is the **flexibility in terms of both the hardware and software**. The second one is that each of the WSAN nodes is always aware of its own structure and can communicate these data to other nodes. On one hand, this provides tremendous possibilities for optimizing node and network operation, and more efficient and cooperative allocation of the tasks between the nodes. On the other hand, this introduces many new challenges in terms of security and data management. We expect that the control theory will have the leading role in solving this issues and developing the respective solutions. In parallel we plan to continue the design and development of the hardware and software components of the platform, as well as continue using it for our experiments. Specifically, our primary focus in respect to the HW development for us now is the design of the new main board built around CC2650 system on chip which is intended to be used in the variety of the ultra-low power applications, including the ones powered with the energy harvested from the environment.

References:


