Testbed Architecture and Framework for Debugging Wireless Sensor Networks

Gabriel Mujica, Jorge Portilla, Teresa Riesgo

Abstract—**The Internet of Things has emerged as one of the key aspects for the future of the Wireless Sensor Networks and their impact on new applications in real environments. This concept poses new challenges in the implementation, testing and debugging of efficient, robust and reliable technologies under this paradigm, specially in a pre-deployment stage where HW-SW platform prototypes are to be optimized prior to their inclusion in actual deployments. In this work, the design and implementation of a complete testbed infrastructure as a support tool for improving the effectiveness and the applicability of sensor nodes to real systems is presented, focused on the modular architecture of the Cookie platform and aiming to help developers to integrate and improve the whole WSN operation to final real-world scenarios.**

Keywords — wireless sensor networks, testbed infrastructure, HW-SW platform, backchannel architecture, node modularity.

I. Introduction

During the last years the paradigm of the Wireless Sensor Networks (WSN) and the Internet of Things (IoT) has been strongly involved in most of the new technologies and research lines for different application fields, such as smart cities, industrial environments, security issues, energy harvesting and home automation, aiming to interact with the environment remotely anywhere at any time. One of the key topics in which the concept of IoT is having more and more impact is in the WSN applications [1], where hundreds of small devices have to be controlled, managed, and monitored in an efficient and reliable way, so that feedbacks from the in-field deployment are to be obtained. However, the challenges that involve the WSN technology, such as power consumption and long-term autonomy, low-data-range based wireless protocols, limited bandwidth, limited processing resource and memory, pose much more issues to be covered in order to effectively expand its applicability to real scenarios.

In this way, the creation of novel testbed infrastructures [2] for experimenting and optimizing WSN technologies must be enclosed by flexible architectures and the ease-of-use on the management, monitoring and reconfiguration of sensor nodes remotely from any place, so that new development approaches and prototypes can be tested. Most of the current testbed platforms are based on commercial hardware which are widely used but start being obsoletes and much less flexible for new technologies that come out with this new world of the WSN and the IoT. Besides, the old node platforms that are used in most of the available testbeds are limited in case of testing new hardware technologies, due to the lack of well-defined hardware debugging capabilities. These issues motivate the creation of a completely new testbed architecture in order to run experiments related to this novel approach within the WSN technology, by continuing developing the Cookies Nodes technology [3] which offers a HW-SW integration platform that accomplishes key features to the future of the WSN within the IoT: Modularity, flexibility, scalability, heterogeneity and repeatability.

The work proposed in this paper is then focused on having a complete feedback to a real Cookie-based WSN by combining a high performance backchannel infrastructure (Based on Ethernet and Wi-Fi connectivity) with the improvement of a memory-segmentation-based architecture to allow users to partially and remotely reprogram their experiments in a dynamic way, by providing a mechanism for modifying specific components of the system. Moreover, new software support components for increasing the robustness of the HW-SW platform are also proposed, in order to help users to debug and experiment in runtime with a real deployment scenario in an easy but reliable way, to optimize their prototypes from the laboratory stage before including them in the final application, so that the efficiency and effectiveness of the nodes can be assured.

II. Related Work

Several testbed platforms have been proposed during the last years, focusing on testing prototypes and applications before final implementations. One of the most well-known testbeds is Motelab [4] that is an open access platform for testing experiments based on MicaZ motes. Nodes can be accessed by a PHP web server. In Kansei [5] the concept of simulation over a testbed support platform is proposed, by implementing 210 sensor nodes, which are connected to Linux-based stargates that control the data collections. SignetLab [6] implements 48 EyesIFXv2 motes that are connected to tiered USB hubs, which are also connected to a single PC in order to control, manage and reprogram the deployed nodes by using a Java based application. In terms of testbeds focused on final application scenarios, different approaches have been presented in recent years, such as WINTeR [7], in which a large scale industrial deployment is proposed in order to evaluate and support industrial applications, based on programmable motes controlled by a web interface. This testbed is based on two backchannels, the first one for controlling the data exchange to the nodes (Ethernet connection) and the second one for powering the nodes (USB connection). In [8] an in-field tracking and localization testbed was developed in a manufacturing lab scenario in order to model the effects that can be found in real deployments with similar conditions, by using TelosB connected to a central server which runs a java-based software application to control the experiments. There are also approaches that propose the management of the testbeds by using the main wireless link, without then implementing backchannel infrastructures (which is bandwidth-consuming and system intrusive when debugging), such as the work presented in [9], where a software component to be installed in the nodes is proposed combined with a web interface to control and manage the deployment using the wireless link.

Most of the current testbed platforms aim to test software prototypes by using the well-known TelosB or MicaZ nodes, which are becoming obsolete due to its low flexibility and their limited adaptation to hardware expansion and testing. Apart from it, the capabilities of dynamic experimentation by performing remote and partial reprogramming of the WSN testbed platform is not covered in most of the available systems, or is only done by means of TinyOS component replacement, which is totally oriented to developments under this operating system.

III. System Architecture – Considerations, Requirements and Challenges

Based on the proposed concept of a WSN Testbed platform which aims to provide developers and researchers with a complete set of HW-SW support tools beyond simulation capabilities, this testbed infrastructure is focused on allowing users to test their prototypes, new algorithms and built-protocols in a real environment, based on a completed pre-deployment scenario where the platform can be programmed and configured with parameters related to the experiment to be performed in runtime. The results of the tests are based on real measurements and the real behavior of the system, not as an interpolation of modeled behaviors, which is the case of WSN simulations, where assumptions are done due to computational and modeling limitations. Users can then access to the parameters of every node, gather actual data of the hardware platform, assess the network performance and, in case of needed, modify configurations in order to change measurement parameters or verify functional blocks. By using the proposed backchannel-based architecture, developers are provided with a way to interact with the sensor nodes and the whole network in a non-intrusive way, in contraposition with the limitations of the main-network-based/backchannel-free testbeds.

In order to face such a challenging goal, four main aspects and considerations have to be covered for the success of the testbed architecture that is being proposed, as split below.

Remote access & reprogramming – One of the key aspects related to the testbed capabilities is the ability to modify not only parameters regarding configurations of the nodes, but also entire algorithms, functional blocks or even the whole application in an remote, efficient and reliable fashion. In this context, the use of an independent channel to replace or download programming files into the core of the nodes without using the main wireless link takes place. The backchannel shall allow modifying the memory flash of the devices where the experiments are running, with no interference with the wireless network itself.

Energy Assessment – one of the research targets in WSNs is the energy efficiency, focusing on extending the lifetime and autonomy of the systems in their final operation. In this way, the testbed infrastructure has to provide the ability to analyze the performance of the nodes in terms of energy consumption in runtime, in order to define the energetic behavior of specific prototypes as well as apply strategies to turn on/off nodes so that the self-reconfiguration of the network in terms of topology changes can be monitored remotely (which is also intended to evaluate implementations such as routing protocols and MAC algorithms). In order to accomplish this, the testbed architecture has to be capable of providing not only strategies to power supply the nodes, but also measurements of the power consumption of the devices remotely whenever it is necessary.

Debugging capabilities – The idea of a WSN testbed platform is not just verify and validate that a sensor is gathering data within a predefined threshold. In order to evaluate the robustness and effectiveness of a WSN, it is necessary to include mechanisms for system debugging, from the hardware layer up to the application level, including the behavior of internal peripherals, interfaces between HW components and software functionalities. Moreover, the interaction among the nodes is also a key aspect to be considered, which introduces the concept of network debugging. Therefore, in this work three levels of system debugging are proposed: hardware level, software-components level, and network-application level.

Multi-experiment – Partial reprogramming strategies – since testing, prototyping and assessing new WSN technologies, algorithms and protocols is an iterative process where optimization are performed based on the results of the debugging tasks, the remote reprogramming needs to be performed in an efficient way so that changes in the experiments shall not imply a long and resource consuming dissemination process every time it is carried out. Therefore, in this work an optimized version of a proposed mechanism based on the concept of partial and remote reprogramming is included, in which several tests units can be downloaded and scheduled into the nodes, so that an experiment repository can be reprogrammed and time slotted within the same device.

Based on these considerations that also define the scope of this work, a novel architecture for testing, debugging and assessing WSN technologies is proposed, focusing on optimizing the performance and efficiency of the Cookie platform with the support of a set of HW-SW developments and components. A general overview of the proposed platform is shown in figure 1 and described in the following sub-sections.

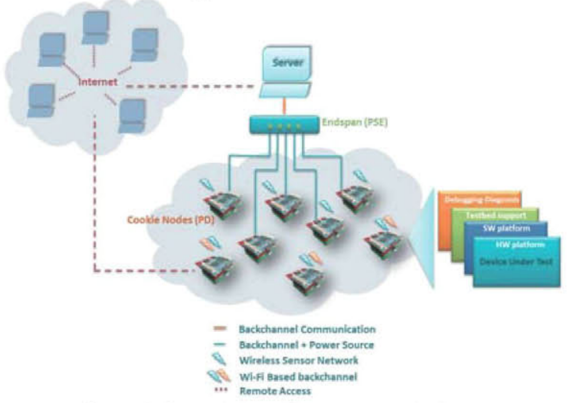

Figure 1. General view of the proposed Architecture.

3.1. Backchannel interface and testbed communication

In the proposed development and testing framework, two main high performance based communication layers are included to satisfy the remote access requirements to the devices under tests. On one hand, a combination of the standard-based Ethernet connection with the concept of Power over Ethernet (PoE) is included by means of designing an integrated Cookie-compliant layer. The idea of using Ethernet is not only related to the reliability of the communication and the bandwidth (apart from its ease of integration in an indoor-based testbed scenario), but also to include the *PoE* technique, which takes advantage of the physical connection of the Ethernet standard to transfer power to end devices by using pairs of wires that are not part of the data signals, so that power supply and data can be obtained from the same cable, then reducing the complexity of the interconnections.

The approach of Power Source Equipments (PSE) shall be also taken into account. These are the devices in charge of acting as both the routers of the Ethernet communication and power supply providers through each available socket. Aiming to have as less complexity in the interconnections as possible, the selected configuration to the PSE has been the use of an Endspan device. Apart from the power supply capabilities of these elements, the switching actions on the device to be powered are done at this level, so the PSE provides with the possibility of turning on/off the nodes remotely.

Finally, the Powered Device (PD) is considered as the final nodes that receive both data signals and powered signals from the PSE through the same cable. As defined in the PoE standard, it is possible to obtain up to 54V and up to 15.4W from the Ethernet port which completely covers the power supply requirements of the end devices. In case of the proposed architecture, the powered devices are the Cookie Nodes, which are the final devices under test. Within the implementation of a Cookie-compliant PoE device, the conversion of the high input voltage levels of the Ethernet port to the standard levels of the Cookie layers is done (3.3V/2.8V, 2.5V, 1.8V 1.5V and 1.2V, for components such as the FPGAs, microcontrollers, among other internal elements included within the system), as well as the inclusion of the measurement of the power consumed by the platform, so that the energy behavior can be analyzed in runtime. Moreover, the Ethernet stack is also included so that the remote actions are transparently performed over the backchannel by using a standard interface, such as SPI or UART, as shown in fig. 2.

On the other hand, to tackle the limitations of the Ethernet in terms of mobility of the sensor nodes as well as the scalability and flexibility of the platform, apart from the performance evaluation in outdoor environments, the inclusion of the IEEE 802.11 within the testbed infrastructure is also proposed, which enhance one of the key features to be covered in these scenarios: interoperability and heterogeneity.

A high performance wireless protocol for performing the backchannel debugging tasks also implies several advantages in the support of final in-field outdoor applications. First, in those scenarios where Wi-Fi connection is limited in certain areas, cluster nodes can be defined as heterogeneous devices that implement the low-rate based communication protocol for the wireless sensor network, and the high performance connection to interface the remote server. On the other hand, in both indoor-outdoor-based testbed scenario, the physical topology in terms of localization and placement can be easily modified, which provides developers with the ability of testing mobility, routing performance, coverage and planning algorithm issues in sensor networks.

Similar to the Ethernet-based Cookie layer, a new communication layer for high level wireless connectivity is then proposed, which includes the integration of a Low-Power Wi-Fi. Hence, the design of a heterogeneous WSN testbed architecture is proposed in order to maximize the capabilities of the remote experimentation schema, where four main communication technologies converge: Ethernet, IEEE 802.11, IEEE 802.15.4 and ZigBee, merging robustness and flexibility with modularity in a unique support platform.

3.2. Hardware support platform – debugging blocks

Unlike works where the main way of debugging remotely the sensor nodes is by performing *printf-based* messages from the node's microcontroller, the present work addresses the capability of carrying out this actions by accessing remotely specific hardware and software elements of the Cookie modular platform, i.e., capturing specific signals of interest such as GPIOs of the node core, sensor interfaces and their status, and communication interfaces among the different layers. Moreover, the ability to analyze specific software components by including a remote debugger for the microcontroller of the platform helps developers to cope with specific failures or application malfunctions that cannot be detected by simple *printf-based* message usage. In this way, bringing the possibility of carrying out runtime debugging tasks by capturing the status of specific registers, memory sectors, flag-masks parameters and peripheral configurations, developers are able to modify the behavior of their experiments, more than just gathering data related to the application running.

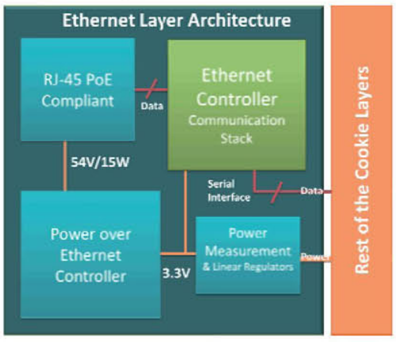
Ethernet Layer Architecture
RJ-45 PoE Compliant
Data
Ethernet Controller Communication Stack
54V/15W
Serial Interface
Data
Power over Ethernet Controller
3.3V
Power Measurement & Linear Regulators
Power
Rest of the Cookie Layers

Figure 2. Ethernet Layer Architecture.

In fig. 3, the proposed debugging scheme is shown in detail, where the FPGA is connected to the majority of digital signals and interfaces involved in the Cookie platform. By combining the hardware blocks implemented in the FPGA with the debugging connection to the microcontroller and, therefore to its internal peripherals, the testbed can have control of both hardware and software elements.

On one hand, the debugging blocks can obtain information related to the digital interfaces of the sensors, such as I2C, 1-wire, PWM, as well as the status of the generic interfaces proposed in [10]. On the other hand, the interfaces related to the connection from the microcontroller to the communication modules can also be gathered and filtered by the FPGA blocks in a transparent way, i.e., without interfering in the behavior of those lines, so that specific signal levels shall be analyzed in order to find low level problems during information exchanging.

Since every Testing Action has its corresponding code number (called Test Identifier, which includes the input information of the testing process received from the testbed server side, as a reference of the action to be performed), the debugging block is in charge of generating the subsequent Action Frames in order to provide developers with the result information according to the response/behavior of the component/functionality under test. The actions to be performed can be classified depending on the type of debugging level, as follows:

- GPIO/Signal Level
- Sensor/Actuator Level
- Interface/Communication Level
- Component/Peripheral Level
- Implementation Level
- Application Level

Every test will automatically generate an Action Frame by the debugging block in the corresponding encoded format, according to the input information of the Test Identifier. The Action Frames are then sent through the proposed backchannel interface.

Regarding the debugging capabilities of the software functionalities related to the microcontroller, it has been necessary to design and integrate a adaptable debugging component in order to remotely perform actions in runtime. In this way, since most of the available development tools support debugging capabilities by using standard interfaces, the remote debugging of the microcontroller is done by means of JTAG, UART or SPI, and the corresponding conversion to the backchannel interface is then carried out in the FPGA, which process is transparent from the user point of view. Thus, by including a small code segment in the uC for the debugging tasks, the capabilities of available IDEs that support the corresponding core (in this case the well-known and extended 8051-based architecture) can be used.

3.3. Memory-segmentation-based architecture

As previously explained, the concept of splitting the available program memory into modular segments to remotely reprogram it in a partial fashion is used, so that different and independent testbed experiments can be downloaded and scheduled to be run at different times. In this way, the novel segmentation-based architecture proposed in [11] has been optimized for the testbed architecture, particularly in the support segment, where the defined software libraries for the framework support are included. The dynamic organization of this area has been optimized to include new software components for the management and control of novel hardware developments. In this way, the area of interest is calculated as follows:

$$A_L = \sum_{n=0}^{Ns-1} (L_{Sn} + O_{Sn}) \quad (1)$$

A_L is the reserved area for the software support platform, which contains the libraries and components to be used. L_{Sn} is the size of the slot n, which depends on the total sub-area of every library in its current version. N_S is the amount of libraries to be included in this area, whereas the O_{Sn} is a generic offset which is assigned to every slot in order to include replacements, modifications or changes in the current version of the libraries. Applying the same offset in every library slot, O_s, the expression remains as follows (note that every offset can also be calculated by assigning a weight depending on the size of the current library):

$$A_L = (O_S * N_s) + \sum_{n=0}^{Ns-1} L_{Sn} \quad (2)$$

Two versions are available, the first one aimed to be used in final implementations where the size of the segment is optimized, not assigning O_s to any sub-area, so the first element of the right side of the expression is null, resulting on a smaller size of the support area. In this case, the target of the experiments is related to the usage of the dynamic area for testing algorithms, applications, etc. on top of the support area, which thus stays fixed. The second version is intended to be applied in platform experiments where modifications of the support libraries are carried out. In this scenario, every available sub-area for library updates is calculated as follows:

$$A_{Sn} = L_{Sn} + O_{Sn} + O_{Sn-1} + O_{Sn+1} \quad (3)$$

A_{sn} is the maximum size per slot that can be updated. Every time a slot is updated with a new version of a library, L_{sn} and O_{sn} have to be updated as well, so that every time a sub-area shall be updated, the adjacent slots will not be corrupted. With this expression the size of the support area is optimized to minimize the insolated memory space, without being necessary to reprogram the whole memory sector, and thus maximizing the capabilities of the partial reprogramming mechanism.

On the other hand, from the point of view of the execution of different experiments downloaded into the testbed platform, the application scheduler has been improved by taking into account the relative position of the trigger services for every user experiment in correlation with the real position of the interruption vector defined in the core architecture. Users can assign time slots for different experiments and the management segment will automatically launch the corresponding application according to the stored information in the programmed scheduler.

3.4. Software support platform – software components

Based on the HW-SW integration platform proposed in [12], a complete set of new software libraries for handling the backchannel debugging tasks has been implemented, so that users only have to configure few parameters in order to run a new experiment into the testbed architecture. More than this, the design and creation of functional blocks for allowing the runtime remote reconfiguration of each node through the laboratory communication were carried out and implemented. These modular components are included in addition to the framework that provides users with libraries for controlling every aspect of the hardware platform, such as communication modules (ZigBee, IEEE 802.15.4) sensor's repository, internal peripherals, etc. as well as the implementation of algorithms and protocols.

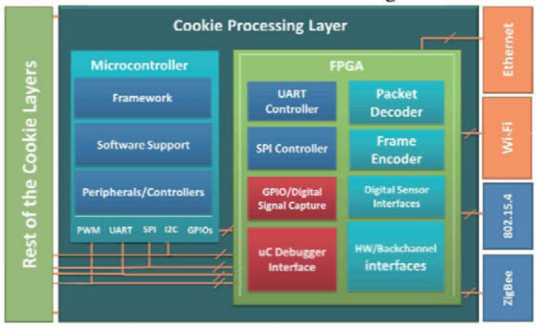

Figure 3. HW-SW debugging schema.

Since the main target of debugging WSN based applications is not only the assessment of every single node but also the network as a whole system, in this work the idea of integrating network analysis for carrying out experimental actions is included, in order to study the performance of the wireless network in terms of connectivity issues, routing protocol robustness, node synchronization, medium access efficiency and platform coverage.

IV. IMPLEMENTATION

In order to address the aforementioned requirements and the proposed system architecture, both hardware and software implementations have been carried out to support the testbed architecture, as described in the following sub-sections.

4.1. Hardware infrastructure

Focused on the definition of the Ethernet-based backchannel communication, two elements are to be considered in the implementation schema. First, the Netgear GS110TP switch [13] has been selected as the Endspan element, which supports the PoE standard as well as 10/100 Mbps (Ethernet and fast Ethernet, respectively). Secondly, the design of the Cookie-based Ethernet Powered Device has been carried out. The design and implementation of this concept has been focused on covering three key aspects of the proposed architecture in a unique device. First of all, a Cookie-compliant design has to be taken into account, which aims to the flexibility, modularity and robustness of the architecture. Second, the platform must be compliant with the PoE standard as Powered Device and thus being able to convert the input levels of the power supply coming from the Ethernet port in order to provide energy to the modular layers that are part of the node platform. Third, an adequate interface between the processing layer of the Cookie platform and the data signals of the Ethernet standard has to be established, so that the node is able to exchange information to the server by using that interface (through standard protocols such as TCP/IP).

The final implementation of the proposed Cookie-Ethernet-based platform layer is shown in fig. 4 (left), called EtherCookie**.** Two stages in the power supply system design have been followed. On one hand, a PoE controller for adapting up to 54 V to 3.3 V has been included in the implementation, the Silvertel Ag9403-2BR from the Ag9400 family [14], in which nominal output voltages can be found, such as 24V, 12V, 5V and 3.3V. For this particular case, the Ag9403 provides 3.3V and 6.6 watts as maximum output power that is enough for the requirements of the Cookie nodes. On the other hand, linear regulators for supplying the different required voltages have also been included in the proposed layer. It has been also necessary to include a RJ-45 PoE-compliant, such as the BelFuse SI-52008-F [15]. In terms of data exchange through the backchannel, a SPI-Ethernet interface solution has been adopted by selecting the WIZNet W5200 module [16]. This module supports most of the standard protocols for Ethernet communication, such as TCP, UDP, IPv4 among others, having High Speed SPI.

Regarding the integration of a Wi-Fi-based node as part of the testbed architecture, a new communication layer for high level wireless connectivity is proposed, based on the RN-131C from Roving Networks. This module addresses IEEE 802.11 b/g as the implemented protocol with different authentication modes such as WEP-64 and WPA2-PSK, typical output RF power = +18dBm,

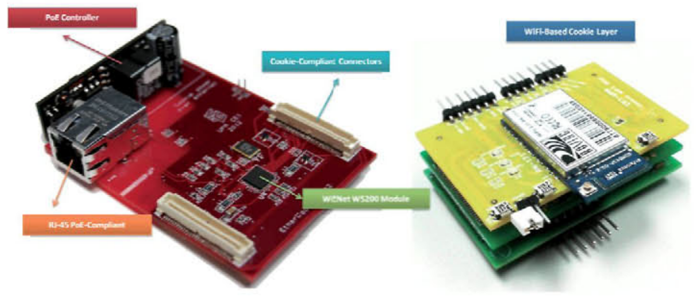

Figure 4. Implementation of the backchannel architecture.

Power Source = 3.3 V, several power consumption modes, having 4 uA in sleep configuration, on-board chip Antenna,, UART interface with AT command-based configuration, and several functionalities such as Timers, GPIOs, 8 Mbit flash memory and 128 kB RAM [17]. The design of the layer follows the standards and dimensions of the Cookie platform, as shown in fig. 4 (right) where the hardware modularity is highlighted.

4.2. Hardware-Software Support Platform

Apart from the existing integration platform, new support components are proposed in this framework to be included in the processing elements, as follows.

CEI_Ethernet: for the configuration and maintenance of the backchannel form the point of view of the EtherCookie implementation. Functional blocks include W5200 configuration, protocol stack configuration and management, which are included in the FPGA hardware implementation.

CEI_WiFi: similar to *CIE_Ethernet* for the support of the testbed backchannel infrastructure, including Access Point and Ad-hoc network configurations, sleeps modes and TCP/UDP connectivity, within the FPGA implementation.

CEI_Reconfig & CEI_Console: These packets include the capabilities regarding remote reprogramming and system recovery through the backchannel interface, in combination with remote options for user segment edition and application code execution based on the proposed optimization scheme.

CEI_Linker: This packet contains information regarding the correlation between the library segments and the user dynamic area, as well as initialization directives to support the optimized segmented architecture.

CEI_Diagnosis: this packet implements the functional blocks regarding deployment analysis and prototype assessment.

CEI_Framework: this packet represents the backbone of the testbed architecture and the support of the platform, which is to be used to create new experiments and application tests as well as assure the effectiveness of the partial remote reprogramming.

V. EXPERIMENTS & TEST CASES

5.1. Scenario1: Backchannel capability-energy performance

In order to test and analyze the performance of the testbed infrastructure, an indoor WSN testbed deployment has been carried out at CEI–UPM, which includes the backchannel interface by using EtherCookie and Wi-Fi-based nodes to remotely access the platform capabilities, as shown in fig. 5.a). Moreover, the idea goes beyond testing the remote interface, because the memory-segmentation-based architecture has been validated and the energy performance of the nodes was analyzed as well. Two processing layers have been included in the experiments, the first one that includes an ADuC841 microcontroller from Analog Devices [18] and the second one with a C8051F930 from Silicon Labs [19]. In this particular test case, the memory segmentation schema was distributed as shown in fig. 5.b), where 40 KB were reserved to the user dynamic area for experimental purposes, dividing this memory space into 4 KB per slot so that up to ten different tests can be downloaded into the reprogramming architecture (which is fully configurable depending on the user requirements). Moreover, from the position 0xB000 to 0xDFFF the support area was also included, which contains the software libraries for providing users which the Cookie HW-SW management capabilities, as well as the recovery segment from 0xE000 for the remote reprogramming and back-up features.

The behavior of the network is as follows: Nodes are configured with the corresponding parameters regarding WSN deployment and the testbed capabilities, and then the basic application starts running (ZigBee-based communication for the wireless connectivity of the nodes). Nodes send information of their sensor measurements every 2 seconds and, after 5 consecutive data transmissions, the nodes are configured to enter in sleep modes. From the point of view of energy modeling and assessment, a power consumption characterization has been carried out by using the on-board circuit included in the Cookie platform for measuring the consumption of the nodes, as shown in fig. 6, where transitions among the different aforementioned stages are highlighted. The first area corresponds to the module configuration (the Wi-Fi-based node in the figure), which takes 1.4s and 2.24s during the connection to an access point of the lab. The average consumption of the ADuC841-based nodes is 68 mA, whereas in case of the C8051F930-based nodes is 40mA. As seen in the figure, there is a time transition between configurations that corresponds to the restart and synchronization process of the Wi-Fi module, which is a common value for the whole deployment. Moreover, current picks regarding packet transmission process are highlighted, which mainly correspond to the power consumption of the communication module.

From the point of view of the remote and partial reprogramming by using the proposed architecture, the tests consisted on sending a whole reprogramming file which included the set of software libraries and packets, and compared it with a partial reprogramming which contained only the top level application

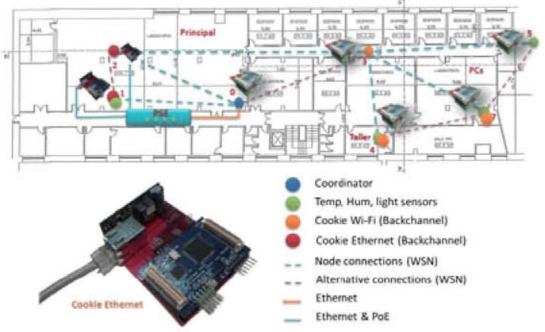

Figure 5.a) Testbed deployment at the research Center.

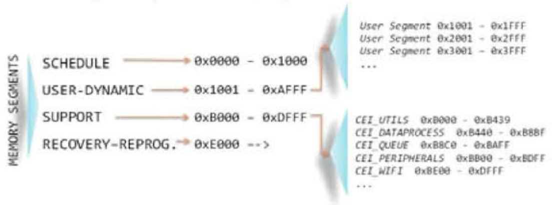
MEMORY SEGMENTS
SCHEDULE
0x0000 - 0x1000
USER-DYNAMIC
0x1001 - 0xAFFF
SUPPORT
0xB000 - 0xDFFF
RECOVERY-REPROG.
0xE000 -->
User Segment 0x1001 - 0x1FFF
User Segment 0x2001 - 0x2FFF
User Segment 0x3001 - 0x3FFF
...
CEI_UTILS 0xB000 - 0xB439
CEI_DATAPROCESS 0xB440 - 0xB8BF
CEI_QUEUE 0xB8C0 - 0xBAFF
CEI_PERIPHERALS 0xBB00 - 0xBDFF
CEI_WIFI 0xBE00 - 0xDFFF
...

Figure 5.b) Memory segmentation schema for the test case.

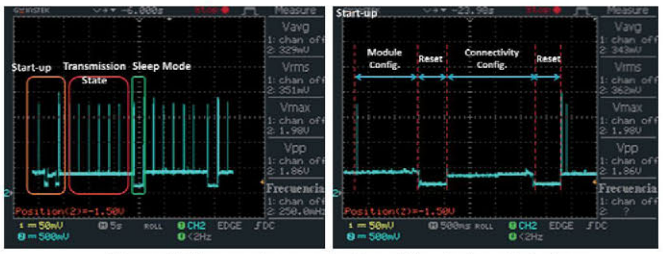

Figure 6. Energy characterization of the deployed platform.

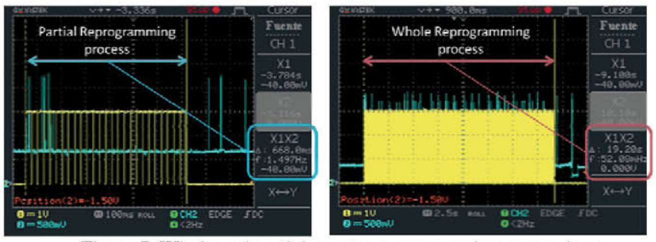

Figure 7. Whole and partial remote reprogramming comparison.

functionalities, considering that the libraries have been pre-downloaded as an integral part of the system. In fig. 7, the remote reprogramming process is compared in both whole and partial reprogramming, obtaining a very important optimization in terms of energy and time consumption during the reconfiguration tasks. In table 1, a comparison between the whole and partial reprogramming regarding file size and timing is done for both processing layers.

Table 1. performance of the remote reprogramming technique.

Remote Reprogramming / Processing Layer	ADuC841		C8051F930	
	Time	Programming File (kB)	Time	Programming File (kB)
Partial Reprogramming	664 ms	1,14	952 ms	1,63
Whole Reprogramming	19,20 s	32,5	21,70 s	36,8

According to the experimental results of the remote reprogramming tests, the time expended to send a single byte remotely is calculated based on the equation 4, taking into account that the main bottleneck in this process is the serial interface between the backchannel communication module and the processing element. For instance, a performed test with the Wi-Fi module for reprogramming the microcontroller through the UART interface configured in 19200 bauds, obtains a value of 570us/byte during the remote reprogramming process, which can be optimized by using higher baud rates or the SPI interface.

$$\mathrm{Pr} = \frac{\mathrm{Tp}}{\mathrm{Sp}} \tag{4}$$

Where Pr is the ratio of reprogramming time spent per byte, Tp is the total amount of time for the partial reprogramming, and Sp is the size of the reprogramming file in bytes.

5.2. Scenario 2: Planning Tool use case–Network analysis

In this experimental test, the main idea is to evaluate and compare simulated deployments of a planning tool with a real indoor environment such as the laboratory area, based on the proposed testbed platform. Three main capabilities of the testbed infrastructure are validated. First, by using the wireless based backchannel, the mobility over the area of interest is proved to be much flexible without reducing the performance of the testbed (in case of a planning tool it is important to carry out an iterative process by changing the position of the nodes in order to compare those deployments with the generated simulations). Second, the network diagnosis techniques have been applied in this particular case in order to obtain real information of the behavior of the nodes in terms of connectivity, quality of the coverage, etc. Third, the remote and partial reprogramming capabilities have also used in order to change application/network set-ups, such as module configurations, power mode strategies, nodes synchronization,

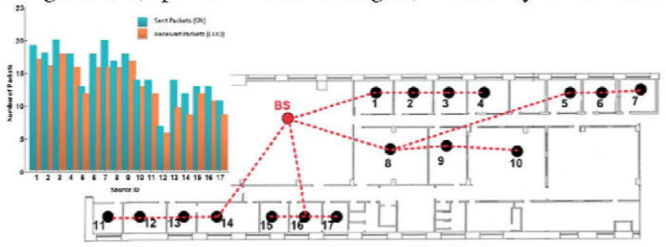

Figure 8. performance of the second testbed scenario.

among other parameters. One of the deployments, composed of 18 nodes, is shown in fig. 8, along with the real routing maps generated by applying the proposed network diagnosis, as well as the path assessment – PLR computation for every single node communication (from node source to the coordinator node), as shown in fig. 8.

VI. Conclusions and Future Work

A complete testbed architecture for debugging, testing, managing and analyzing Cookie-based wireless sensor network prototypes and new developments has been designed and implemented, not only aiming to have a robust and reliable laboratory support tool for experiments, but also to increase the efficiency, connectivity and remote control of the wireless nodes in order to provide users with an effective way to interact with the platform anywhere at any time, hence joining key points to the future applications in this research field. As future approaches for continuing potentiating the usability of the proposed testbed architecture, one of the key features to be implemented is a service oriented GUI, so that users can access and control the testbed in a more intuitive way through web-based internet connectivity. Moreover, as the HW-SW platform has been created for being scalable, the deployment of more amounts of nodes along the facilities of the research lab is possible and feasible, so the next step is to increase the number of nodes connected to the testbed infrastructure.

References

[1] S. Agrawal, M.L. Das, "*Internet of Things — A paradigm shift of future Internet applications*", Nirma University International Conference on Engineering, NUiCONE 2011. pp. 1-7.
[2] S. Krco, M. Nati, D. Pfisterer, N. Mitton, T. Razafindralambo "*A survey on facilities for experimental internet of things research*" in IEEE Communication Magazine, Vol. 49, pp. 58-67. November 2011.
[3] J. Portilla, A. de Castro, E. de la Torre, T. Riesgo, "*A Modular Architecture for Nodes in Wireless Sensor Networks*" in JUCS, vol. 12, nº 3, pp. 328-339, March 2006.
[4] G. Werner-Allen, P. Swieskowski, M. Welsh, "*Motelab: a wireless sensor network testbed*" in Information Processing in Sensor Networks, IPSN 2005, pp. 483 – 488.
[5] E. Ertin, A. Arora, R. Ramnath, V. Naik, S. Bapat, V. Kulathumani, M. Sridharan, H. Zhang, H. Cao, M. Nesterenko, "*Kansei: a testbed for sensing at scale*" in Information processing in sensor networks, IPSN 2006, pp. 399–406.
[6] R. Crepaldi, S. Friso, A. Harris, M. Mastrogiovanni, C. Petrioli, M. Rossi, A. Zanella, and M. Zorzi, "*The design, deployment, and analysis of signetlab: A sensor network testbed and interactive management tool*" in TridentCom 2007. pp. 1 –10.
[7] J. Slipp, C. Ma, N. Polu, J. Nicholson, M. Murillo, S. Hussain, "*Winter: Architecture and applications of a wireless industrial sensor network testbed for radio-harsh environments*" in Communication Networks and Services Research Conference, CNSR 2008. pp. 422 –431.
[8] M. Bal, H. Xue, W. Shen, and H. Ghenniwa, "*A testbed for localization and tracking in wireless sensor networks*" in Systems, Man and Cybernetics, SMC 2009. pp. 3581 –3586.
[9] T. Dimitriou, J. Kolokouris, N. Zarokostas, "*Sensenet: a wireless sensor network testbed*" in ACM MSWiM 2007, pp. 143–150.
[10] J. Portilla, T. Riesgo, A. Abril, A. de Castro, "*Rapid prototyping for multi-application sensor networking*", 12 November 2007, SPIE Newsroom. DOI: 10.1117/2.1200711.0851.
[11] G. Mujica, V. Rosello, J. Portilla, and T Riesgo, "*On-the-fly dynamic reprogramming mechanism for increasing the energy efficiency and supporting multi-experimental capabilities in WSNs,*" in Proc. IECON'13, pp.5455-5460, Nov. 2013.
[12] G. Mujica, V. Rosello, J. Portilla, T. Riesgo, "*Hardware-software integration platform for a WSN testbed based on cookies nodes*", in 38th Conference on IEEE Industrial Electronics Society (IECON'12). pp. 6013-6018.
[13] Netgear PoE, h*ttp://www.netgear.com/business/products/switches/*
[14] Silvertel PoE products,*http://www.silvertel.com/poe_products.htm*
[15] Bel Fuse components, *http://www.belfuse.com/*
[16] WIZNet korea, TCP/IP Chip, *http://www.wiznet.co.kr/*
[17] Ultra-low power Wi-Fi RN131C module, *http://www.rovingnetworks.com/products/RN131C.*
[18] Analog Devices, *http://www.analog.com/en/index.html.*
[19] Silicon Laboratories, www.silabs.com