SmartEye: An energy-efficient Observer Platform for Internet of Things Testbeds

Riccardo Pozza
Centre for Communication System Research
University of Surrey, Guildford
GU2 7XH, Surrey, UK
R.Pozza@surrey.ac.uk

Alexander Gluhak
Centre for Communication System Research
University of Surrey, Guildford
GU2 7XH, Surrey, UK
A.Gluhak@surrey.ac.uk

Michele Nati
Centre for Communication System Research
University of Surrey, Guildford
GU2 7XH, Surrey, UK
M.Nati@surrey.ac.uk

ABSTRACT

The recently growing need to experiment with Internet of Things (IoT) technologies in more realistic environments requires the experimenter to have remote and precise observation of heterogeneous IoT devices under test. This paper introduce *SmartEye* an energy efficient observer platform for IoT testbeds. *SmartEye* embeds many features that are currently available on different observer system into a common platform, while placing energy efficiency for autonomous operation at the core of its design. It will provide the foundation for a new class of IoT testbeds to be deployed in more realistic testbed environments such as cities, urban areas or more remote deployment environments without compromising on the rich observational and management features provided by today's observer boards.

Keywords

SmartEye, Testbeds, Sensor, IoT, Platform, Observer, Management, Performance

1. INTRODUCTION

The difficulties associated with bringing research on advanced Future Internet technologies into real world deployments have sparked a recent trend in the research community towards the use of testbed to experimentally evaluate research outcomes in realistic environments. Evidence thereof provide the GENI community in US or the FIRE community in Europe, which work on the creation of heterogeneous testbeds that cover a wide spectrum of enabling technologies that will play an important role in the Future Internet. The IoT is such a technology and different testbed for it have started to emerge [4], in which wireless sensor network (WSN) nodes play the most prominent role. While most IoT testbed installations have been initially confined to lab environments, however outdoor testbeds have started to recently emerge. Examples thereof are testbed deployed on roof tops [9, 2] of a building or even spanning areas of a city [11, 8].

In contrast to service specific IoT deployments which comprise of the actual hardware that is necessary to support a particular service or set of services (e.g. wireless sensor nodes, gateways devices), testbeds can incorporate additional infrastructure - so called observer boards that provide advanced observational and management capabilities related to the individual nodes of an IoT testbed and the software components that may run on them. These observer boards are often colocated to the IoT nodes and directly attach to them via multiple I/O interfaces to exchange information out of band, take measurements of hardware behavior or provide experimentation specific stimuli. Typical functions performed by such observer boards are the remote reprogramming of IoT nodes for a particular experiment, the out-of-band collection of debugging information or calculated KPIs, the sending of control commands to change the behavior of experiments, measurements of energy consumption, injection of real world events or interference etc.

Depending on the nature and diversity of management and observation capabilities they provide, observer boards are likely to be based on platforms that exceed the capabilities of the attached IoT nodes they observe, i.e., embedded Linux computers. Testbeds that employ such observer boards are primarily lab-based testbeds in indoor environments or located outdoors attached to a lab building and designed with the assumption that power supply is readily available. However there is an increasing need for such observer nodes to be deployed *out-in-the-wild* where the continuous availability of energy sources cannot be guaranteed (e.g. solar panel, intermittent availability of street light power). This requires the reconsideration of the design of existing observer boards to take into account energy constraints, allowing them to become more autonomous in their operation.

In this paper, we present *SmartEye* an energy efficient observer platform for heterogeneous device platforms in IoT testbeds. *SmartEye* embeds many features that are currently available on different observer system into a common platform, while placing energy efficiency for autonomous operation at the core of its design. *SmartEye* will provide the foundation for a new class of IoT testbeds to be deployed in more realistic testbed environments such as cities, urban areas or more remote deployment environments without compromising on the rich observational and management features provided by today's observer boards. Our contributions can be summarised as follows.

We report on the design and implementation of an energy efficient observer platform for IoT testbeds which supports multiple IoT devices and provides a rich set of observation and management capabilities to cover the most common use cases for IoT experimentation. Our initial design evaluation concentrates on WSN nodes as an important IoT device class. Energy efficiency is achieved by a careful selection of platform components, modular design and additional hardware features on the platform that allow fine grained energy management depending on a specific usage scenario. The observer platform provides an extensible design that allows interfacing additional sensors and communication modules, that can be made available to attached IoT nodes during experimentations to extend their capabilities for a particular experiment. Furthermore, we provide a detailed quantitative evaluation of the proposed observer platform focused on its energy consumption for different usage scenarios and a detailed characterization of the energy breakdown over its components. We benchmark our platform to other existing observer platforms in terms of energy consumptions and capabilities and prove in principle its self-sustainable operation.

The remaining paper is structured as follows. Section 2 surveys related work in the area. An overview of the *SmartEye* and motivating requirements are provided in Section 3, while Section 4 details the design of its hardware and software components. Section 5 provides a detailed evaluation of *SmartEye*. Final conclusions are provided in section 6.

2. RELATED WORK

Attaching observer platforms to IoT and wireless sensor network (WSN) testbeds has been a feature commonly used in testbed installations. Early examples of these observation boards can be found in lab- or building-scale testbeds such as MoteLab and TWIST [4]. In such scenario, the observers are mainly represented by Linux computer providing physical USB connectivity to a number of WSN nodes and connecting them to a central testbed server through a wired Ethernet connection. Initial platforms where primarily limited to testbed management tasks, showing only limited emulation and measurement capabilities.

In order to further enhance their capabilities, external custom *emulation boards* have been designed for working with some specific sensor platforms. Exploiting the USB connection, the *EnvEmu* [13] board can be connected between a TelosB and its observer, allowing to change the sensor perceived value of temperature, light and humidity and thus emulating the occurrence of a given phenomenon. In addition, battery exhaustion or some sort of battery replenishment through specific energy harvesting technique [3] can also be emulated. Other boards such as the *Test Daughter Board* developed within the Senslab [10] testbed go beyond providing emulation capabilities. Coupled to a standard management board it is able to monitor all the GPIO pins exposed by the device under test, thus allowing not just injecting events but also profiling a device energy consumption and I/O activity. A similar solution for a TelosB mote is *Aveksha* [12] which represents a Telos Debug Board (TDB). It operates as sister board for a TelosB mote and provides JTAG control and debug, and through it energy monitoring and execution profiling. Although these solutions can improve the observations capability of original observer, most of the times they represent platform specific solutions that also always need a counterpart board in order perform other testbed management tasks, such as reprogramming and collecting out-of-band statics.

Despite the many possibilities such combination of observation and emulation boards can give for the interaction between the testbed management tools and the attached IoT platform, all these functionalities come at a cost: the additional energy required for powering such observer. This limits the applicability of the aforementioned solutions only to indoor testbeds or situations where mains power is readily available.

In order to overcome the boundaries of the lab and to move experiments to the wild, a set of custom and more integrated boards have been recently designed in order to cope with the constraints that such environment poses, such as the lack of wired connectivity to the back-end server and unlimited power resources. *SeeDTV* [7] represents the first attempt to move testbed observation and management outside of the lab. As extension board for MicaZ motes, it features an embedded LCD screen that can provide immediate visual feedback in response to some predefined observation and management commands manually raised to the connected platform and without the need to be connected to an additional board, but still missing the possibility to be remotely managed at real-time. Differently, *FlockLab* [2] provides the first example of a custom observer built around a an embedded Gumstix Linux board and a standard WiFi module. Other than supporting basic testbed management functions, the FlockLab observer platform integrates additional capabilities such as onboard storage and observation ability for 4 different devices through the use of specific interface slots that provide access to the platform GPIO pins and means for voltage control and power measurement. While integrating most common management and observation functions for outdoor testbed devices, FlockLab still misses the target to be an energy efficient observer board, due to the consumption of its components. A first attempt to provide an ultra low-power observer board is represented by [14]. *DualMote* represents one of the current ongoing approaches aiming at coupling two standard motes through a UART connection in order to use the first one for managing and monitoring a second mote that actually acts as device under test.

The price of enabling very energy efficient and low power observation of a testbed device is the lack of bandwidth for sending and receiving debug information and management tasks, due to the use of low-power transmission modules based on IEEE 802.15.4. The observation and interaction capabilities among the observer and the testbed device are limited due to the use of a UART channel and a 8MHz microcontroller. An approach similar to that provided by the DualMote platform has been realized also in the SmartSantander [11] testbed, where a *Waspmote* platform [6] has been enhanced with a second Xbee radio devoted to receive management messages such as over-the-air reprogramming images to be tested. The effectiveness of the observation capabilities of such approach is limited by the employing a single 8MHz Atmel microcontroller shared among testing and managing tasks, which also impacts the capabilities to assess the true system performance of a protocol on the target MCU.

In order to overcome the lack of existing energy efficient wireless management solutions for outdoor testbed devices, some solutions have also been designed exploiting energy harvesting techniques for their autonomy in outdoor deployment. Platforms such as the *CitySense* [8] node and the *Meshlium* [6] gateway deployed in Santander are able to self-sustain during the day by scavenging energy from the mains power of street lights during nighttime.

3. SYSTEM OVERVIEW

This section provides an overview of the *SmartEye* platform and the underlying requirements that have motivated its design.

3.1 Requirements

One of the main considerations has been the ability of the observer system to operate in an outdoor environment, without constant availability of a mains power source, while providing a rich set of capabilities and features required for the most common IoT experimentation use cases. Running purely on battery is not feasible for longer periods of time without maintenance effort. As observer systems need to operate on demand based on changing experimenter needs, duty cycling is difficult to realise without loosing flexibility. Autonomy of the system should be achieved by the use of intermittent power sources such as solar panels or opportunistic power available during the periods of operation of a street light. The overall aim was to reduce the energy consumption of the system as much as possible, while providing sufficient capabilities in terms of processing power, sampling rates required for measurements and communication bandwidth towards the testbed infrastructure.

A second important design consideration was the support of heterogeneous IoT device platforms by the observer system.

A third requirement deals with the modularity of the platform design. In order to make the platform suitable for multiple testbed deployment environments support for a diversity of communication modules and types of sensor / actuators is highly desired. However putting all onto one board may quickly lead to unnecessary hardware costs and heavily impact the available energy budget on the board. It is therefore important that the observer board can be easily customized by putting together a required configuration of communication and sensing modules for a particular deployment scenario.

The above consideration primarily focused on non-functional requirements of the platform. In addition, there are various functional requirements that an observer board for most common IoT testbed use cases must fulfill [4]. We can briefly group the desired functions according to the *observational capabilities* and *sensor event emulation*, *communication capabilities*, *management capabilities* and *node capability extensions*. The following list highlights the ones considered important for the *SmartEye* design:

- Monitoring of GPIO pins of attached IoT nodes
- Energy consumptions profiling of attached IoT nodes during operation with high resolution (> 1KHz) including / excluding the USB power consumption
- Injection of emulated sensor events via DACs to the DC inputs of IoT nodes

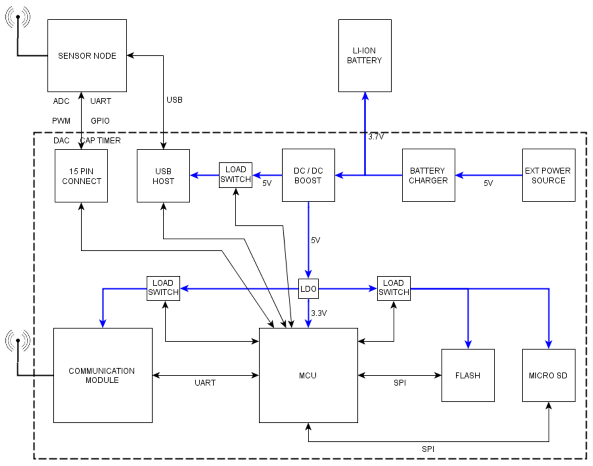

Figure 1: System Architecture

- Ability to exchange information directly with the IoT nodes for the purpose of receiving debug information or performance statistics or relay commands for experimentation control
- Wireless streaming of live debug/statistics information during experiments to the testbed backbone, even for a meshed multi-hop setup with other observer boards
- IoT node power provision including controlled power off and reset
- Remote reprogramming by flashing images required for experiments on request to IoT nodes
- Local storage of collected debug information or performance statistics locally or images for IoT nodes prior reprogramming

3.2 System architecture

This section provides an overview of the overall architecture of the system realized as shown in Figure 1. Based on the aforementioned requirements, we have devised a system architecture consisting of the main functional blocks necessary for the realization of the *SmartEye's* system.

At the heart of *SmartEye* is a microcontroller unit (MCU) which executes the firmware for the *SmartEye* observer platform. It processes all requests received from experimenters via the control and management plane of the testbed and coordinates resulting activities across all peripherals handling the required observation, emulation and communication tasks.

Another important role of the MCU is to perform advanced energy management functionalities to minimise the energy consumption across the platform. As energy efficiency is an important design consideration, load switches are added to all main subsystems of the observer platform. This means the IoT node related subsystems, communication subsystems as well as storage subsystem can be individually powered down by the MCU, if they do not require involvement in a particular system activity. The load switches thus provide more fine grained energy management control for a more energy efficient operation of the system.

In order to support the attachment of diverse IoT nodes, our choice was to go for a USB Host functionality as many of today's mote class devices provide USB connectivity, e.g. via an FTDI chip. This allows the bi-directional communication with the attached IoT node as well as the bootstrap loading of images.

For the measurement of the energy consumption of attached IoT nodes, energy measurement circuits are connected via ADC to the MCU. Our system embeds an energy measurement circuits directly attached to the USB port that provides power to the IoT node. Moreover, an additional circuit can be attached behind the USB circuit of an IoT node, so the energy consumption of an IoT node without its USB subsystem can be measured. The observation subsystem towards the IoT nodes also boasts various GPIO lines and ADC channels as generic observation channels that can be attached to the IoT node to capture more fine-grained information about an IoT node's performance, actuation decisions and internal states. In addition DAC lines are provided which can be attached to ADC channels of the IoT node, providing user controlled sensor emulation capabilities via the MCU.

The communication subsystem has been designed with modularity in mind, so different configurations of *SmartEye* for particular deployment environments can be created in an resource efficient manner. Most of today's wireless communication modules such as WiFi, Bluetooth, GPRS modems, *ANT+* or 802.15.4 offer UART connections. Our design allows the connection of one or more wireless communication modules, including an Ethernet module (not described further here) for wired integration into an indoor testbed.

The storage subsystem allows to trade-off energy efficiency with storage size. An external flash provides cheap and energy efficient storage for smaller amount of data, while for larger volumes of data an external micro SD card can be utilised.

Finally, the power supply circuit is composed of a battery charging unit capable to convey energy from an external source such as a battery charger or a solar panel to a Lithium Ion Battery. The battery was chosen for its compactness and better efficiency as well as their vast available in many mobile phones around the world making them affordable at low cost. In order to support the USB Host interface a 5V output voltage is needed to comply correctly to USB standard specifications. For this purpose, a DC/DC step up converter had to be integrated in the design in order to provide the 5V power rail. The 5V rail is then downconverted to a 3.3V regulated output via a low dropout LDO regulator, which distributes the power to other subsystems of the board, including MCU, memory and communication.

4. DESIGN AND IMPLEMENTATION

This section will in detail describe our concrete choices in terms of hardware (HW) component and software (SW) design for the realisation of the proposed *SmartEye* observer system.

4.1 Hardware

In order to achieve a modular design with respect to the communication subsystems, we decided to separate out the communication modules from our main board. In our preliminary proof-of-concept realisation we have built two communication daughter boards, one for Ethernet and one for WiFi. Figure 2 shows the main board with the MCU and other aforementioned subsystems and the WiFi communication board. Our current prototype still uses various discrete components, which will be replaced by surface mount components for subsequent production.

One of the key design consideration for *SmartEye* was the choice of the MCU. There is a wide range of energy-efficient MCU platforms available for embedded devices ranging from the 8bit ATmega128 family, the popular 16bit MSP430 family or a large set of ARM based 32bit MCUs. While the aforementioned 8 or 16bit architectures can provide higher energy efficiency with many fine grained MCU power states, their limitations in computational capabilities and clock speed, built-in peripherial and memory makes them less attractive for our demands. As we like to support concurrent sensor emulation and live-streaming of multiple high sampling rate observation channels back to a testbed backbone via multi-hop communication with potentially multiple communication interfaces a 32bit MCU seems to be a more adequate choice. Some 32bit MCU also come with a wide range of peripherals such as multiple UARTS and SPIs, ADC and DAC unit as well as built in Ethernet support and USB Host functionality, which can minimise the amount of external peripherals required and increase the flexibility of connecting external devices for future extensions. After a careful investigation, we decided that the NXP LPC1768 which is based on an ARM Cortext M3 architecture provides the best tradeoff between required processing power (up to 125MIPS at 100MHz), available peripherials and energy consumption (4 reduced power modes up to 517nA). Many of the MCU pins have been brought externally on our port and can be configured by software to to be UARTs, ADC inputs, DAC output, Timers capture inputs, GPIOs and PWMs, thus enabling interaction with external communication devices and observation and emulation services with the attached IoT nodes. For the MCU, our board offers two separated crystal oscillators, a 32.768KHz for the Real Time Clock and a 12MHz for the Core PLL.

In order to complement the USB Host interface provided by the MCU, two ferrite beads have been added to the power supply lines to avoid ESD events by the devices plugged into the USB bus. An ultra low capacitance double rail to rail ESD protection diode from NXP provides additional signal protection from damages caused by electrostatic discharges and other transients in the USB differential data lines pair. In addition to that, a power measuring circuit was included featuring a Texas Instrument's INA138 high-side shunt current measurement device which measures the current provided by the USB Host +5V rail to the USB attached Device. The output signal of this circuit was fed to a Microchip MCP603 buffering rail to rail operational amplifier and finally to an analog input connected to the internal LPC1768's 12 bit ADC.

For the design of the storage subsystem a comparison between different memory elements was made. Given the requirements of low cost and low power consumption, a small but low cost 16Mbit serial flash Atmel AT25DF161 was chosen, which offers an SPI interface for programming and reading and requires only 5uA in Deep Power Down. If additional stroge is needed, a microSD card could be added to expand the onboard storage capabilities. A respective microSD card holder and circuit have been added to the board.

As aforementioned we realised an Ethernet and a WiFi

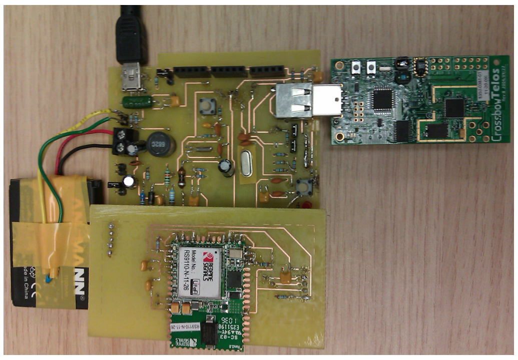

Figure 2: SmartEye

daughter board for the communication subsystem. While the Ethernet board is used for mainly indoor deployment scenarios where power availability is not an issue (i.e. Power over Ethernet), the choice of the right WiFi communication module represented a more important design consideration. As can be seen from our evaluation in the Section 5 the overall energy consumption of the observer board is dominated by the WiFi communication module. In addition as the WiFi module can operate at the same time as the radio of the IoT node, different frequency bands should be ideally selected in order to avoid interference. We therefore considered different ultra-low power WiFi modules available that can operate in the 5GHz band. We narrowed down our choices to two available modules on the market, the Silex SX-SDCAG 802.11a/b/g SDIO card module and the RS9110-N-11-26 from Redpine Signals. In the end we selected the Redpine Signals module for its possibility to adopt lower power modes states (up to 4mA in standby) as well as its capability to run at 5GHz mode both in Ad-Hoc and Infrastructure mode providing also a full TCP/IP stack.

The realisation of the power circuit requires a battery charging circuit, which is based on the Texas Instrument's BQ2057 advanced linear charge management integrated circuit capable of exploiting a 5V power supply rail to recharge a 4.2V end-voltage Lithium Ion Battery with an with end charge current of 500mA. A National Semiconductor's LM2621 low input voltage step up DC/DC converter provides us with a 5V power rail for the USB Host. While supporting up to 1A of load currents, it features a low shutdown current of less than 2.5μA. The last part of the powering circuit is the LDO regulator that steps down the voltage to the 3.3V rail which powers all the other components of the board. Load switches used for powering off the different subsystems are based on the the Fairchild Semiconductor FPF2123. The selected load switch operates with a shutdown current of less than 2μA and provides at the same time protection to eventually connected devices on the USB bus, which may encounter large current conditions. Switch control is performed by a signal directly controlled by a the GPIOs of the MCU.

4.2 Software

In the following we briefly describe the development of the software components that run on top of the MCU of the *SmartEye* board and the PC side testing tool that we developed for control and validation of the envisioned *SmartEye* functionality. It should be noted that the purpose was to build a proof-of-concept to validate in principle all envisioned functions of the board. A more comprehensive firmware which will integrate all features smoothly with our testbed control and management plane as well as smart energy management is planned for a second release this summer.

Our firmware for the board is written in C and is based on the NXP LPC17xx CMSIS (Cortex Microcontroller Software Interface Standard), which provides for an HW abstraction layer for the ARM MCUs and includes start up files and predefined interface for accessing hardware on a basic level. Our prototype includes drivers for communication with all subsystems and application logic to perform the desired observation, emulation and management tasks.

A USB Host Controller driver was written by adapting the NXP USB Host lite driver to correctly recognize and configure attached IoT nodes. Our implementation currently supports all platforms based on an FTDI chip. We used the Microsoft USBView software to sniff out appropriate driver configurations. By an appropriate study of the TelosB mote datasheet and an FTDI driver implementation for linux, it was also possible to build a driver on top of the USB Host driver to enable the bi-directional communication with an attached mote. Furthermore, a driver for the MSP430 bootstrap loader was made to enable erasing, resetting and reprogramming the sensor node's flash. The current implementation thus supports communication with all devices with an FTDI chip. The same approach for driver design can be followed in order to support Sunspot motes and to demonstrate the flexibility of SmartEye to support other platforms.

A driver for the WiFi module was implemented, exploiting UART routines to interact with it and the setup of TCP/IP sockets with a connected laptop in Ad-Hoc mode. In addition, a driver for the external flash memory was written to obtain routines with SPI Commands to erase, program and read the memory.

In order to validate all implemented HW/SW functions of the observer board, a PC side control and testing tool was developed which is shown in Figure 3. It supports all main functions such as establishing a remote connection from the observer board, intialisation of different subsystem components, management functions such as reseting of nodes, upload and bootstrap loading of images the IoT nodes, bi-directional transfer of commands and debugging data to and from the IoT nodes as well as observational capabilities such as the retrieval of energy consumption readings or other observation made by the ADC of the *SmartEye* board. In the next section we will describe different test cases that we conducted in more detail.

5. EVALUATION

The following section provides an evaluation and assessment of the energy efficient features of the *SmartEye* observer platform. We first provide a qualitative discussion of the supported capabilities, benchmarking *SmartEye* against other state-of-the platforms. We then perform a detailed quantitative study into the energy efficiency of our platform in order to evaluate the goodness of our design choices. Finally, we examine the case of autonomous operation of *SmartEye* in case of outdoor deployments, identifying under what conditions self-sustainability can be achieved.

5.1 Platform Capabilities

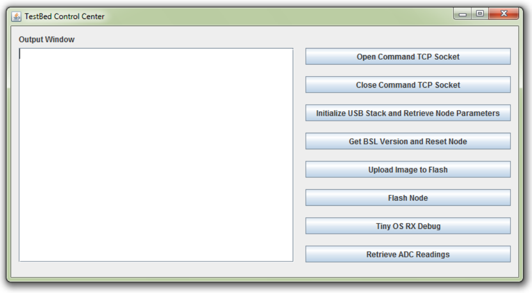

Figure 3: PC side control and testing tool for the *SmartEye* platform.

Based on our analysis of related work (Section 2) the following three platforms have been selected as representative of observers for outdoor testbed experiments: *FlockLab*, *DualMote* and *CitySense*. Table 1 provides a comparison of the different observer platforms. As those platforms are not commercially available and no accurate design specifications are available for their reproduction, we confine our comparison to a qualitative one based on available information and data sheets. It shows that the *SmartEye* platform has the best coverage in terms of required features (see Section 3) for observer boards. *DualMote* and *CitySense* represent platforms designed for outdoor testbeds made of specific IoT testbed devices. In contrast *SmartEye* and *FlockLab* are designed to support a wider range of IoT devices. Differently from *FlockLab*, SmartEye embeds a USB Host controller, which allows it to be extended to support all IoT platforms designed with a USB interface. Our initial prototype demonstrates support for TelosB motes. Using the same approach for driver design other platforms featuring the same FTDI USB chip (e.g. iSense, SunSpot or Waspmote) can be easily supported.

All observer platforms provide wireless communication capabilities, which is instrumental for outdoor deployments. However the low data rate radio of *DualMote*, despite being ultra-low power, makes it inadequate to support reliable real-time control and streaming of high resolution observational data of the IoT device under test. The support of the IEEE 802.11*n* standard makes *SmartEye* and *Flocklab* particularly suitable for deployment in WSNs and IoT testbeds as they can avoid interference when operating concurrently with an experiment in the 5GHz band.

The table also roughly compares the energy efficiency of the different observer boards, by considering only the most power hungry components (MCU and wireless radio) from the data sheets against the real measured maximum consumption of *SmartEye* (please see next section for more details). As can be seen from the table *DualMote* provides the highest energy efficiency from all platforms. However the ultra-low power design sacrifices both data rate and computational power, leaving most desired features unfulfilled. *SmartEye* in contrast fulfills all desired features for the most common use cases of an observer board, yet has a significantly lower current draw than both *Flocklab* and *CitySense*, thus allowing our board to easier operate autonomously as discussed in Section 5.3. A final important note concerns the very low component price of the *SmartEye* board, making it more suitable for larger testbed deployment when compared with the other boards.

Table 1: Existing Observers comparison

Feature	SmartEye	FlockLab	DualMote	CitySense
Platforms	4+	4	1	1
Mgmt.	✓	✓	✓	✓
Observation	✓	✓		
Emulation	✓	✓		
Connectivity	802.11n	802.11bgn	802.15.4	802.11abg
Autonomy	✓		✓	✓
Cons.(mA)	234.8	760+	27.7	1400+
Cost($)	<100	>170	$\approx$ 100	>200

5.2 Energy Consumption Breakdown

For this set of tests we used a standard bench Black-Star 3225 MP multimeter to evaluate the current breakdown among the different components of the *SmartEye* board. Every measurement is taken for a stable output and repeated 10 times before averaging the final result. In order to determine the consumption for the MCU in varying operating modes, we measured the current flowing in the entire board after the 3.3V LDO regulator, while keeping all the peripherals switched off. We then measured the current flowing in the WiFi module, in the Flash memory on various test setups and lastly in an TelosB node both from 5V and from 3.3V. The corresponding results are shown in Table 2. The MCU consumption is evaluated in Deep Power Down, Power Down, Deep Sleep, Sleep, Running at 10MHz and Running at 100MHz, showing that the device is capable of trading-off between performance and power consumption, thus enabling energy efficient behavior. As for the WiFi module, the consumption is measured in the following modes: sleep, connected to access point as well as receiving and transmitting in Ad-Hoc mode while connected to a laptop. The results show that the WiFi module is capable of being put in low power modes thus reducing the power consumption when connected but not transmitting. Furthermore, we measured with the MCU's timers the time needed by the WiFi module to turn on and setup a connection. This could be accomplished in only 428ms, showing the possibility to turn off the device and save further additional energy when not

Table 2: Breakdown of Current Consumption

Component	Power Mode	Current
MCU	Deep Power Down	790μA
MCU	Power Down	11.595mA
MCU	Deep Sleep	11.598mA
MCU	Sleep	28mA
MCU	Running at 10MHz	13.609mA
MCU	Running at 100MHz	51.8mA
WiFi	RF, Baseband and Core Sleep	10.293mA
WiFi	Connected to AP	183mA
WiFi	RX/TX in Ad-Hoc mode	193.5mA
Flash	On, idle	26μA
Flash	Programming	4.47mA
Flash	Reading	4.5mA
Flash	Erasing Chip	12.532mA
TelosB	USB + Non USB	87.2mA
TelosB	Non USB	32.2mA

needed. The consumption for the Flash Memory are evaluated when in idle mode, programming and reading as well as when erasing the entire chip, showing a generally low power consumption. Finally the IoT node TelosB in an idle configuration showed a major consumption at 5V because it includes also USB components such as the FTDI in its supply circuit, whereas if powered at 3.3V from the battery pins the consumption showed is far less.

Apart from measuring the current breakdown over the different components, we also evaluated the consumption for various test scenarios that resemble typical use cases of operation while experimenting in a testbed. During this tests measurements are taken from the battery charger at 5V. The corresponding results are shown in Table 3. Scenario 1 resembles the best case, where the MCU is in Deep Power Down mode and all the peripherals are switched off. Scenario 2 and 3 represent the worst case in which the USB device is powered and configured, the WiFi module is connected and either receiving or transmitting. As data is streamed via WiFi to the testbed backbone the flash memory is assumed to be turned off. The scenarios represent either the case of streaming live debug messages from the sensor node to testbed or the case of management/continuous emulation during which control messages are uploaded to the platform from the testbed framework. Scenario 4 is the offline debugging, where after the task of collecting data in the flash, the upload of its 2MB entire content to a central server is made through the WiFi module. In this case, an evaluation of the duration of the task was possible and made showing in the best case 22647ms necessary to read the entire flash and to correctly transmit it on a TCP socket to the server running on a laptop. Nevertheless this figure is only indicative since it's dependent on both the software implementation and the transmission protocol. Scenario 5, instead shows the reprogramming of a node image stored in the external flash. For this purpose we used an image of ROME [1] which fills up nearly the entire ROM available ($\approx 48KB$) for a TelosB mote. We measured the time needed for the reprogramming was 17069ms. Finally, scenario 6 represents a energy profiling case in which the ADC is performing sampling of the current at the maximum speed of 200Ksample/s. However, since this sampling frequency would generate a large amount of data, a reading of the ADC's result registry was made every 500μs thus producing only 24KB/s and not losing accuracy in terms of profiling. In this case, the WiFi is off and the only agents active are the USB node, the MCU and the external Flash. Figure 4 shows an example of realtime energy profiling including USB of the ROME protocol. The blue line shows the current consumption of a node alternating between awake and asleep states with a duty-cycle d equal to 0.1, while the red one shows a node always awake and with the radio in listening mode ($d = 1.0$).

5.3 Self-Sustainability Assessment

The results of the last section provide us with an understanding of energy consumption for both typical and worst cases of operation of *SmartEye*. In this section we assess the self-sustainability of the platform by providing an approximate estimate of the dimensions of a photovoltaic cell solar panel for typical and worst case operation. Starting with the worst case consumption, *SmartEye* requires $290mA$, leading

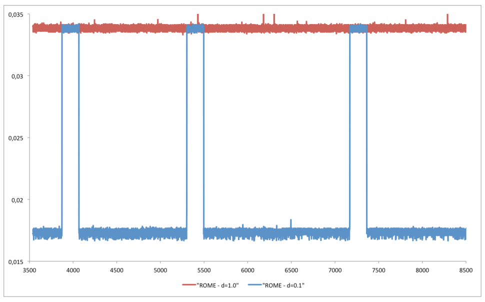

Figure 4: ROME Power Profile (Current(mA) vs Sample)

Table 3: Test Scenarios

	Scenario	Current
1	Power Down	5.44mA
2	Realtime Debug	290mA
3	Realtime Control	290mA
4	Offline Debug	271mA
5	Reprogramming	105.3mA
6	Energy Profiling	160mA

to a total power consumption of:

$$P = V \times I = 0.29 \times 5 = 1.45W$$

This means that daily, the consumption is quantifiable in:

$$P_D = P \times 24h = 1.45W \times 24h = 34.8Wh$$

We queried the Photovoltaic Geographical Information System database [5] provided by the Joint Research Center Institute for Energy and Transport using their PV estimation utility to accurate dimension the solar panel based on real solar illumination data. We used as the location the our University campus where we aim to deploy *SmartEye* in an outdoor testbed (51°14′59" North, 0°34′0" West, Elevation: 30 m a.s.l). We assumed optimal inclination and orientation of the panel and an estimated loss of 10% due the inefficiency of the energy accumulation device. Considering an estimated loss due to angular reflectance effects and due to local temperature data leads to a combined loss of 19.7%. Performing some trial with different values of peak power (Wp) of crystalline silicon panels, we found out that the worst case is in general during the month of December, as it can be seen in Table 3. Moreover, assuming the board is continuously operating for 24h/day in the worst case mode, the required daily average of at least 35Wh can be sustained by at least a 45Wp panel during the December month. However the costs for such panel are rather high. Furthermore dimensioning the panel to the worst case will result in energy being wasted in the remaining part of the year. For example, the year average daily production is of 115Wh which is more than 3 times of what is needed and in summer this will rise to a 176W in July which is more than 5 times the energy required. In a more typical scenario of utilization, the usual activity will involve a Reprogramming of the nodes, the Energy profiling for some time and the offline upload of the data back to the server. Since the reprogramming and the upload task are made through a reliable and fast link such as WiFi, the most important contribution to consumption is given by the Energy Profiling task. This accounts for a 160mA (of

Table 4: Test Scenarios

Month	Monthly kWh	Daily kWh
Jan	1.42	0.046
Feb	2.15	0.077
Mar	3.28	0.106
Apr	4.70	0.157
May	5.31	0.171
Jun	5.15	0.172
Jul	5.45	0.176
Aug	5.05	0.163
Sep	3.85	0.128
Oct	2.78	0.090
Nov	1.71	0.057
Dec	1.08	0.035
Year Average	3.50	0.115

which 87.2mA are for powering the TelosB) and since an 8h daily full utilization would be an appreciable reference for a Testbed use, this shows that a 6.4Wh daily energy is needed, which is easily provided by 4Wp solar panel, that is a tenth of the worst case which would produce a year average daily energy of 11.5Wh which is roughly two times the need.

6. CONCLUSION AND FUTURE WORK

This paper presented *SmartEye*, a new hardware platform designed to enable low-energy observation of large scale and outdoor WSNs and IoT testbeds. A detailed evaluation of its consumption shows that *SmartEye* is more energy efficient than other platforms with similar observational and management features and thus able to effectively perform typical testbed observation and management tasks in a self-sustainable way, e.g. by just relying on solar panels as a power source. This makes the platform deployable in most outdoor environments. The already very low energy consumption of the *SmartEye* platform can be further reduced based from the initial experiences gained by our evaluations. A revision of the platform is currently under development, which will form the basis of an IoT outdoor testbed deployment consisting of 100 *SmartEye* observer boards. The availability of a such large outdoor testbed operated through the *SmartEye* platform will give use the possibility also to better characterize and assess the effectiveness of our platform in a real operating scenario.

7. REFERENCES

[1] S. Basagni, M. Nati, C. Petrioli, and R. Petroccia. Rome: Routing over mobile elements in wsns. In *Proceedings of the 28th IEEE Conference on Global Telecommunications*, GLOBECOM'09, pages 5221–5227, Piscataway, NJ, USA, 2009. IEEE Press.

[2] J. Beutel, R. Lim, A. Meier, L. Thiele, C. Walser, M. Woehrle, and M. Yuecel. The flocklab testbed architecture. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 415–416, New York, NY, USA, 2009. ACM.

[3] P. De Mil, B. Jooris, L. Tytgat, R. Catteeuw, I. Moerman, P. Demeester, and A. Kamerman. Design and implementation of a generic energy-harvesting framework applied to the evaluation of a large-scale electronic shelf-labeling wireless sensor network. *EURASIP J. Wirel. Commun. Netw.*, 2010:7:1–7:12, Feb. 2010.

[4] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo. A survey on facilities for experimental internet of things research. *Communications Magazine, IEEE*, 49(11):58 –67, november 2011.

[5] JRC. Institute for energy and transport (iet). http://re.jrc.ec.europa.eu/pvgis/apps4/pvest.php.

[6] Libelium. http://www.libelium.com/products/.

[7] H. Liu, L. Selavo, and J. Stankovic. Seedtv: deployment-time validation for wireless sensor networks. In *Proceedings of the 4th workshop on Embedded networked sensors*, EmNets '07, pages 23–27, New York, NY, USA, 2007. ACM.

[8] R. Murty, A. Gosain, M. Tierney, A. Brody, A. Fahad, J. Bers, and M. Welsh. CitySense: A vision for an urban-scale wireless networking testbed. In *Proceedings of the 2008 IEEE International Conference on Technologies for Homeland Security, Waltham, MA*, 2008.

[9] NITOS-Testbed. http://nitlab.inf.uth.gr/NITlab/index.php/testbed.

[10] sensLAB. http://www.senslab.info.

[11] Smartsantander, 2010. http://www.smartsantander.eu/.

[12] M. Tancreti, M. S. Hossain, S. Bagchi, and V. Raghunathan. Aveksha: a hardware-software approach for non-intrusive tracing and profiling of wireless embedded systems. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pages 288–301, New York, NY, USA, 2011. ACM.

[13] L. Tytgat, B. Jooris, P. D. Mil, I. Moerman, and P. Demeester. Demo abstract: Wilab, a real-life wireless sensor testbed with environment emulation. *6th European Conference on Wireless Sensor Networks, (EWSN'09)*, pages 2–3, 2009.

[14] H. Wu, J. Cao, X. Liu, and Y. Liu. Dual-mote: A sensor network testbed for high rate sensing-transmission and runtime evaluation. In *Wireless Communications and Networking Conference (WCNC), 2011 IEEE*, pages 2048 –2053, march 2011.