OpenMote: Open-Source Prototyping Platform for the Industrial IoT

Conference Paper · September 2015

DOI: 10.1007/978-3-319-25067-0_17

4 authors:

Xavier Vilajosana
Universitat Oberta de Catalunya
131 PUBLICATIONS **2,399** CITATIONS
SEE PROFILE

Pere Tuset-Peiro
Universitat Oberta de Catalunya
34 PUBLICATIONS **640** CITATIONS
SEE PROFILE

Thomas Watteyne
National Institute for Research in Computer Science and Control
160 PUBLICATIONS **4,003** CITATIONS
SEE PROFILE

Kristofer S. J. Pister
University of California, Berkeley
305 PUBLICATIONS **18,431** CITATIONS
SEE PROFILE

OpenMote: Open-Source Prototyping Platform for the Industrial IoT

Xavier Vilajosana[1], Pere Tuset[2], Thomas Watteyne[3], and Kris Pister[4]

[1] Universitat Oberta de Catalunya, Spain
xvilajosana@uoc.edu
[2] OpenMote Technologies, Spain
pere.tuset@openmote.com
[3] Inria Paris-Rocquencourt, EVA team, France
thomas.watteyne@inria.fr
[4] UC Berkeley, EECS, USA
pister@eecs.berkeley.edu

Abstract. This paper introduces OpenMote, the latest generation of Berkeley motes. OpenMote is a open-hardware prototyping ecosystem designed to accelerate the development of the Industrial Internet of Things (IIoT). It features the OpenMote-CC2538, a state-of-the-art computing and communication device. This device interfaces with several other accessories, or "skins", through a standardized connector. The skins developed to date include boards to provide power, boards which enable a developer to easily debug the platform, and boards to allow seamless integration of an OpenMote network into the Internet.
This hardware ecosystem is complemented by a suite of software tools and ports to popular open-source IoT implementations. The OpenMote platform is for example tailored to run the OpenWSN open-source implementation of emerging IIoT standards. The combination of hardware and software ecosystems gives an embedded programmer an intuitive and complete development environment, and an end-user a fully working low-power wireless mesh networking solution running the latest IIoT standards.

1 Introduction

Tomorrow's Smart Factory will be wireless. Industrial process monitoring and automation applications are both "going wireless" and "going IP" to reduce installation cost and simplify Internet integration. Standardization is leading this effort, for example through the IETF 6TiSCH working group [1], which is standardizing tomorrow's "Industrial Internet of Things" (IIoT).

Early experimentation is needed for these standards to be widely adopted, and for the Industrial IoT to take off. Key accelerators for this process are open-source hardware and software projects which provide early access to implementations of those standards, and link pioneering ideas to industrial adoption. In the early 2000's, this happened through the IEEE802.15.4 [2] standard, the TelosB

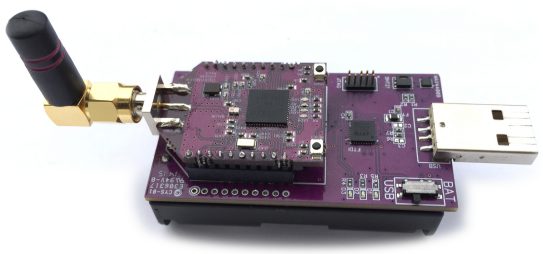

Fig. 1. An OpenMote-CC2538 and an OpenUSB.

hardware platforms and the TinyOS [3] implementation. This combination triggered significant research on Wireless Sensor Networking, resulting in standards such as 6LoWPAN.

This experience has thought us several lessons. First, tight coupling and parallel evolution between hardware platforms and open-source projects benefit the adoption of the standards they implement. Second, a modular hardware design improves the applicability of the hardware to different applications. Third, providing easy-to-use board support packages (BSP) and prototyping tools speeds up time-to-deployment and eventually time-to-market. Fourth, open hardware benefits knowledge transfer and industrial adoption, as companies can take advantage of already proven designs. Fifth, symbiotic alignment between standardization groups and open-source hardware/software projects yields better standards and speeds up their adoption. These lessons learnt where the basis when designing the OpenMote hardware ecosystem.

This paper introduces OpenMote[1], a modular open-hardware ecosystem designed for the Industrial IoT. The OpenMote platform is designed to efficiently implement IIoT standards such as IETF 6TiSCH. It was designed within Berkeley's OpenWSN [4] open-source project, and is therefore perfectly suited for the new wave of IIoT standards such as IEEE802.15.4e TSCH and IETF 6TiSCH. It is an open platform, given users "bare metal" access to state-of-the-art hardware, with current work being done to use it within several additional open-source IoT communities such as Contiki [5], RIOT [6] and FreeRTOS[2].

The remainder of this paper is organized as follows. Section 2 presents other open-source experimentation platforms, and relates them to OpenMote. Section 3 introduces the OpenMote hardware ecosystem and presents the OpenMote platform and its interfaces. Section 4 introduces the tools and software developed around OpenMote. Section 5 presents some results about the performance of the OpenMote hardware. Section 6 reviews use cases and success stories developed using the OpenMote. Finally, Section 7 concludes this paper.

[1] `http://www.openmote.com/`
[2] `http://freertos.org/`

2 Other Open-Source Experimentation Platforms

The OpenMote is the latest in generation of low-power wireless platforms, and adopts their most useful features and follows "lessons learnt".

The "Berkeley motes" were born with the Smart Dust project in 1997. The MICA family was the first one to establish the idea of a small low-power wireless featuring communication, computation and energy. The widely popular TelosB platform was a milestone "Berkeley mote", which combined an open design, state-of-the-art hardware (in 2004), ease-of-use and commercial availability. After more than a decade of lifetime, the hardware it offers is no longer state-of-the art. Compared to today's off-the-shelf solutions, the TelosB lacks memory space, speed and hardware acceleration for security, while consuming more energy.

During the "TelosB decade" (2004-2014), several companies adopted the open hardware design of TelosB, and developed updated versions of it. This includes the TMote Sky, IRIS and Zolertia Z1 motes. Other designs departed from the TelosB constrained design and developed more powerful motes. This includes Sun Microsystems' SunSpot (which embeds a Java virtual machine), the Arduino, or the COU motes [7]. These platforms are targeted mainly at educational use, and lack the reliability and low-power operation required for industrial applications.

Most chip vendors have now switched to 32-bit microcontroller architectures (e.g. ARM's Cortex-M series), which offer more computational power for a lower-power consumption. Systems-on-Chip (SoCs) are available which combine a microcontroller and a radio in a single chip. These SoCs reduce complexity and costs of new designs, while offering higher performance and lower power than their equivalent 2-chip solutions.

The OpenMote is the latest generation "Berkeley mote". It is designed to capture this exciting state-of-the-art technology, while maintaining the simplificity and elegance of a platform such as the TelosB.

3 The OpenMote Hardware Ecosystem

A prototyping platform is not just a single communicating "mote"; it must also encompass accessories it can plug into (including sensors), and tools to help firmware development. This section presents this hardware ecosystem.

The driving idea of the OpenMote ecosystem is to separate the communication/computation module from interface boards, resulting in a simple, modular and elegant solution. The OpenMote-CC2538 (Section 3.1) is the heart of this ecosystem, and provides computation and communication capabilities. Its standardized pin-out enables it to interface to the other elements of the ecosystem using digital and analog interfaces (GPIO, I2C, SPI, UART), through high-level connectors such as Ethernet, USB, Phidgets and Grove sensor connectors. The boards the OpenMote-CC2538 can interface to today include the OpenBattery (Section 3.2), the OpenBase (Section 3.3) and the OpenUSB (Section 3.4).

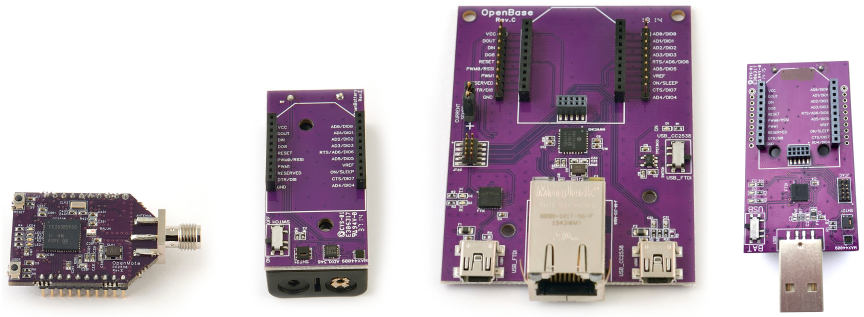

Fig. 2. The OpenMote hardware ecosystem: (from left to right) OpenMote-CC2538, OpenBattery, OpenBase, OpenUSB.

3.1 OpenMote-CC2538

The OpenMote-CC2538 (Fig. 2) sits at the core of the OpenMote hardware ecosystem. It is the brain of the platform, and the element a developer programs. The first generation OpenMote, the OpenMote-CC2538, features a TI CC2538 SoC. The design of the "OpenMote" is, however, generic and future revisions can integrate other SoCs, possibly featuring different communication technologies.

The CC2538, at the core of the OpenMote-CC2538, is a SoC from Texas Instruments with a 32-bit Cortex-M3 microcontroller and an IEEE802.15.4-compliant radio. The microcontroller has a clock speed up to 32 MHz, embeds 32 kB of RAM and 512 kB of flash memory, and features several peripherals (including GPIOs, ADC, I2C, SPI, UART and timer modules). The radio operates in the 2.4 GHz band and is fully compliant with the IEEE802.15.4-2006 standard.

The power subsystem is driven by a step-down DC/DC converter (TPS62730) with two operational modes: bypass and regulated. In bypass mode, the DC/DC converter directly connects the input voltage from the battery (typically 3 V) to the system. In regulated mode, the DC/DC converter regulates the input voltage down to 2.1 V. The benefit of such approach is that the efficiency of the system can be improved under both low and high load conditions (when the system is sleeping or when the radio is transmitting/receiving).

A 32 MHz crystal clocks the radio, and has a drift of up to 30 ppm (parts per million) from -20 C to +70 C. This crystal remains off when the radio is asleep. To achieve tight time synchronization – a fundamental requirement of new industrial communication protocols – a second 32 kHz crystal clocks the microcontroller's RTC (Real Time Clock). This ultra low-power RTC allows the OpenMote to keep track of time, even when in deep sleep. This second crytal is rated at 10 ppm from -40 C to +85 C (the industrial temperature range).

The OpenMote-CC2538 board features 4 LEDS, 2 programmable buttons, a chip antenna and an SMA connector for an external antenna. The form factor and pin-out is the same as other popular low-power wireless board, such as the

XBee and WaspMote. This means that an OpenMote can interface with any accessory built for those boards, and can be a swap-in replacement.

The OpenMote-CC2538 is the core of the OpenMote hardware ecosystem. The modular design separates it from different development interfaces – "skins" – to provide a versatile set of tools to the developer. This design also enables a user to replace today's OpenMote-CC2538 with future versions of the board.

3.2 OpenBattery

The OpenBattery (Fig. 2) is a skin for the OpenMote-CC2538 which provides power and basic sensing capabilities. It is composed of a battery holder for 2 AAA batteries, a socket for the OpenMote-CC2538, an on/off switch, and three sensors: a temperature/humidity sensor (SHT21), a 3-axis accelerometer (ADXL346) and a light sensor (MAX44009). All sensors are interfaced with the OpenMote-CC2538 using an I2C bus. The temperature sensor (and updated version of the one on the TelosB) can be used in a wide set of applications, including network synchronization [8]. The 3-axis accelerometer can be used for dynamic or static motion detection. The light sensor can be used for a wide range of applications, from presence detection to touch-less switching.

3.3 OpenBase

The OpenBase (Fig. 2) is a skin for the OpenMote-CC2538 which offers all the interfaces needed for efficient firmware development. It features a socket for the OpenMote-CC2538, a 10-pin JTAG connector for in-circuit debugging of the OpenMote-CC2538, a circuit to monitor the current draw of the OpenMote-CC2538, pins to interface the OpenMote-CC2538 to external devices, a USB connector to re-program and debug the OpenMote-CC2538, and a 10/100 Mbps Ethernet connector[3] to connect the OpenMote-CC2538 directly to a LAN.

This wealth of interfaces means that the OpenBase can serve several purposes. Through the JTAG interface, it can be used during code development to place breakpoints and inspect variables. Through the USB interface, it can be used to reprogram the OpenMote-CC2538 with pre-compiled binary images, and receive status information from that firmware over a serial interface. Through the 10/100 Mbps Ethernet interface, the OpenMote-CC2538 can be connected to the Internet without requiring a computer.

3.4 OpenUSB

The OpenUSB (Fig. 2) is designed for ease-of-use by ends users, and for using OpenMote-CC2538 boards in a testbed. It features a USB "male" connector, a

[3] The Ethernet connector is based on a Microchip ENC28J60 chip and a standard RJ-45 connector that includes both the magnetics and the circuit protection.

10-pin JTAG connector, a battery holder for 2 AA batteries, the same 3 sensors as the OpenBattery, and a Grove connector[4] to connect to dozens of sensors.

An end-user uses the OpenUSB much like he/she uses a TelosB today: reprogram the board with precompiled firmware, debug either through `printf` statement or through the JTAG interface, and deploy battery-powered nodes. In the contex of a testbed, the OpenMote-CC2538 connected to an OpenUSB (see Fig. 1) is a drop-in replacement of a TelosB. Similar to a TelosB, it can be connected to a small single-board computer (such as the Raspberry Pi) which can reprogram it. This makes it an ideal solution for a testbed.

3.5 Interfaces and Accessories

One of the main requirements for an OpenMote is to be able to interface to other devices and boards. Thanks to its form-factor, numerous "shields" already exist, for example to the Arduino.

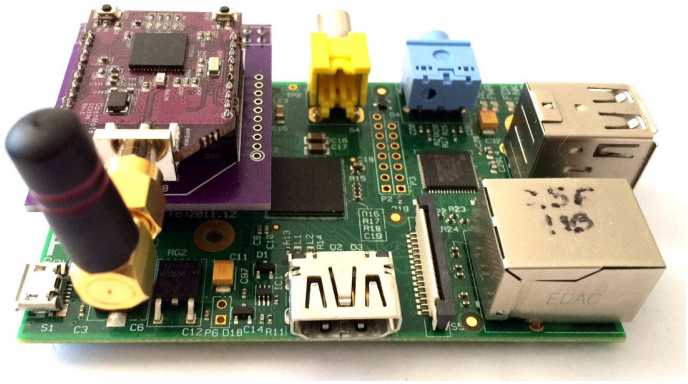

Fig. 3. The Raspberry Pi adapter for the OpenMote.

As part of its continuous push to expand the OpenMote ecosystem, the OpenMote team has developed an adapter board for the Raspberry Pi version 1 and version 2 (see Fig. 3). With this setup, the Raspberry Pi can be programmed with the OpenPi image[5], an OpenWSN-ready distribution for the Raspberry Pi, turning the Raspberry Pi+OpenMote into the gateway of an OpenWSN network.

4 The OpenMote Software Ecosystem

The OpenMote hardware ecosystem is empowered by the OpenMote software ecosystem, a collection of tools to simplify development (Section 4.1) and ports

[4] `http://www.seeedstudio.com/`

[5] `https://github.com/openwsn-berkeley/openpi`

to popular operating system (Section 4.2). Section 4.3 further details the end-user experience of using OpenWSN on the OpenMote ecosystem.

4.1 Tools

The OpenWSN project contains an Eclipse-based development environment for the OpenMote-CC2538, including JTAG debugging. The OpenWSN build system contains the necessary scripts to upload pre-compiled binaries onto the board, when in-circuit debugging is not needed.

The OpenMote community has developed firmware to turn an OpenMote-CC2538 into a IEEE802.15.4 packet sniffer[6]. When connected to an OpenBase (resp. OpenUSB), the OpenMote-CC2538 publishes captured packets onto the Ethernet (resp. serial) interface. In both cases, packets can be analyzed using Wireshark, a popular packet analysis software.

4.2 Operating Systems

The OpenMote team believes in community-driven open-source hardware and software.

OpenMote can be seen as the hardware spin-off of Berkeley's OpenWSN project [4]. A number of OpenMote prototypes were developed in that project to identify the components which are most suitable for implementing IIoT standards such as IEEE802.15.4e TSCH. OpenWSN promotes the Industrial Internet of Things by providing an open implementation of standards such IEEE802.15.4e TSCH [9], IETF RPL [10], IETF CoAP [11] and the standards promoted and developed by the IETF 6TiSCH working group [12].

It is also possible to use the OpenMote with other open-source projects such as FreeRTOS, RIOT and Contiki. The latter has adopted the OpenMote as its prototyping platform through Thingsquare, Contiki's commercial offering.

The OpenMote community has also developed the first low power implementation of the Distributed Queuing (DQ) protocol [13], demonstrating that the concept of DQ can be implemented on off-the-shelves hardware.

4.3 End-User Experience

The OpenMote team aims at providing an integrated hardware/software platform for end-users to use cutting IIoT standards.

For a developer, it enhances user experience by facilitating tasks such as debugging, having access to GPIOs and hardware interfaces leveraging the burden of bare metal programming. When developing on OpenWSN, OpenMote provides all the necessary tools to develop an application or contribute to a protocol implementation. The OpenBase and the OpenUSB provide external pins to debug peripheral buses such as SPI, UART and I2C, using a logic analyzer.

[6] `https://github.com/OpenMote/firmware/tree/master/projects/ieee802154-sniffer`

For an end-user, reprogramming an OpenMote-CC2538 with the latest release of the OpenWSN stack is as simple as running a single command line:

```
scons board=OpenMote-CC2538 bootload=/dev/ttyUSB0 oos_openwsn
```

One OpenMote-CC2538 can be connected to a Raspbarry Pi running the OpenPi distribution (Fig. 3) to turn it into the gateway. The other motes can be connected to OpenUSB boards (Fig. 1) and deployed in the field. The out-of-the-box experience is that the Raspberry Pi becomes the 6LoWPAN "Low-power Border Router" (LBR), connecting the OpenWSN network of OpenMotes to the Internet.

5 Performance

This section presents performance results measured on the OpenMote-CC2538.

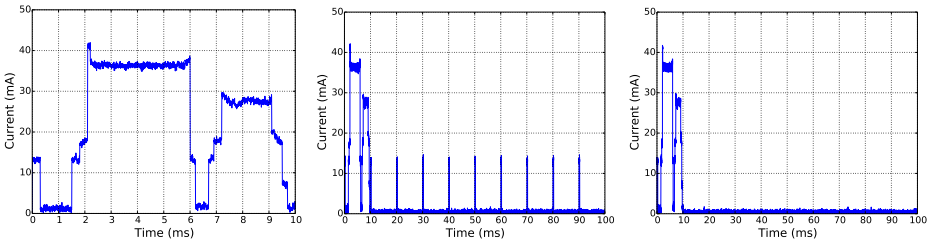

Fig. 4. Current draw of the OpenMote-CC2538 running the OpenWSN protocol stack. (left) The 10ms active slot; the mote transmits and receives an acknowledgment. (center) The active slot followed by 9 inactive slots; the mote wakes up at each slot, the default OpenWSN behavior. (right) Same result, but the node stays asleep during inactive slots (optimization).

Current Draw. To measure the current draw of the OpenMote-CC2538, we program two boards with the latest OpenWSN firmware, and connect both to a computer using OpenBase boards. We configure one node to be DAGroot, the other a regular node.

The data is acquired using a Rigol DS1000E digital oscilloscope and a uCurrent Gold current probe. The current probe is connected in series to the OpenBase current sense pins, and transforms the current flowing into the OpenMote-CC2538 board into a voltage through a low-noise operational amplifier circuit. The current probe is connected to the digital oscilloscope, which acquires and digitizes the analog voltage. The oscilloscope has a vertical resolution of 8 bits and the vertical range is set to 10 mV/div, which yields 195 uA/LSB. The sampling frequency is set to 1 MHz, the acquisition time to 10 ms. Results are show in Fig. 4.

In an active slot, the node transmits a data packet to the DAG root and waits for the acknowledgment (ACK) packet from the DAG root. The data packet is 127 bytes long (≈4 ms duration); the ACK is 32 bytes long (≈1 ms). In an active slot, the transmitting mote waits 1.5 ms, then turns on its radio to transmit the data packet. 200 μs after the data packet is fully transmitted, the mote start listening for the ACK.

The OpenMote's CPU is clocked by a 32 MHz external crystal, rather than the 16 MHz internal crystal. When the CPU is on, the OpenMote-CC2538 consumes 13 mA.

During packet transmission and reception, the CPU consumes around 1.5 mA. This consumption adds up to the radio transceiver current consumption. When transmitting at +7 dBm, the OpenMote-CC2538 consumes 34 mA. When receiving a signal at -50 dBm, it consumes 20 mA. These values match the datasheet.

RF Signal. We capture the transmission of a packet on a spectrum analyzer to measure the power radiated by the radio front-end. Results are shown in Fig. 5. The signal is well centered, demonstrating the good performance of the radio front-end.

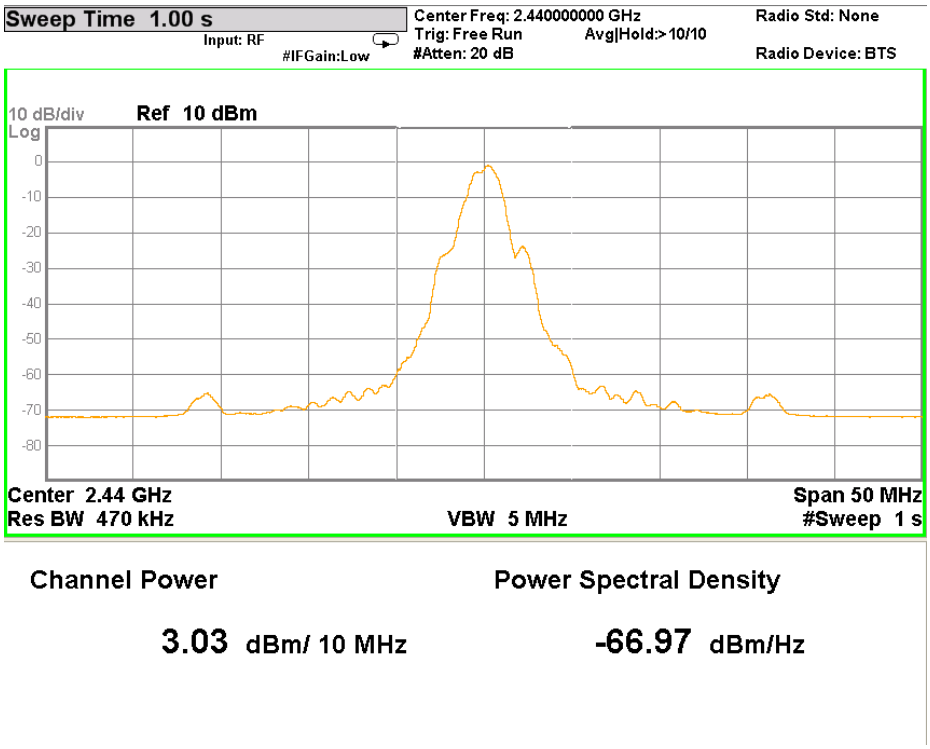

Fig. 5. Power spectral density of OpenMote-CC2538 transmitting at 2.44 GHz.

6 Example Use Cases

This section contains examples of projects in the industrial, standardization, robotics, medical and mobile network fields where the OpenMote is used today.

Industrial. In [**?**], the authors build a energy consumption model of the OpenMote-CC2538 running the OpenWSN protocol stack to predict the networks battery lifetime in critical industrial environments. In [15], the authors study the dependability of low-power wireless networks, and use the OpenMote-CC2538 as the state-of-the-art hardware for industrial applications. In [16], the authors study the applicability of the OpenWSN implementation running on the OpenMote-CC2358 to run wireless control loops in industrial applications

Standardization Support. Standardization of communications protocols is a fundamental step to enable massive adoption of technologies. The initially fragmented IoT communication ecosystem is converging towards several IoT standards, most of them with a common network layer. The IETF is leading this effort by bringing IP to different link-layer technologies, including IEEE802.15.4 and Bluetooth. The IETF 6TiSCH working group is standardizing a protocol stack which combines the industrial performance of IEEE802.15.4e with the ease of use of IP.

In this complex process, the OpenMote helps accelerating the adoption of these emerging standards. Most open-source operating systems, including OpenWSN but also Contiki and TinyOS are adopting 6TiSCH technology and taking advantage of the tools provided by the OpenMote platform. Recently, ETSI organized an interoperability event around 6TiSCH technology [17] in which the OpenMote-CC2538 running OpenWSN was selected are the reference device to test interoperability against.

Robotics. Grieco *et al.* [18] survey the interaction between the fields of robotics and IoT, and identify the OpenMote-CC2538 as the state-of-the-art platform for this type of applications.

Medical. In [19], the authors implement a system capable of real-time on-demand monitoring of patient in hospitals using the OpenMote-CC2538.

Mobile Networks. Weekly *et al.* [20] uses the OpenMote-CC2538 in indoor environmental sensing applications using networks of mobile sensors.

7 Conclusion

This paper introduces the OpenMote prototyping environment. The goal of this modular, versatile, open-hardware platform is to support development and prototyping of tomorrow's Industrial IoT communication technologies. OpenMote has been designed to address the lessons learnt in the past decade of IoT research and early development. OpenMote is supporting and collaborating with most open-source initiatives and standardization efforts around the IoT, including Berkeley's OpenWSN project – the birthplace of the OpenMote – , but also Contiki, RIOT and FreeRTOS.

The hugely popular OpenMote has become the de-facto low-power wireless experimentation platform. Its very active community is constantly supporting new technologies and building a wider hardware ecosystem for the IIoT.

Acknowledgments

The development, testing and commercialization of the OpenMote has been a community effort. The authors would like to thank Kevin Weekly, David Burnett, Mark Oehlberg, Qin Wang and Tengfei Chang for participating in the design and testing of what became the OpenMote, to Ariton Xhafa and his team at Texas Instruments for their help with the CC2538 platform, and to the people at the UC Berkeley Swarm Lab for their constant support. We also thank the members of the OpenWSN community and the Thingsquare team for providing the requirements of the platform from a user point of view. A special thanks to the hundreds of OpenMote customers for believing in open-hardware for the Industrial IoT.

References

1. D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, "6TiSCH: Deterministic IP-enabled Industrial Internet (of Things)," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 36–41, December 2014.
2. *IEEE802.15.4-2011: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE Computer Society Std., Rev. IEEE Std 802.15.4-2011, 5 September 2011.
3. P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An Operating System for Sensor Networks," in *Ambient Intelligence*. Springer Berlin Heidelberg, 2005, pp. 115–148.
4. T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, "OpenWSN: A Standards-Based Low-Power Wireless Development Environment," *Transactions on Emerging Telecommunications Technologies (ETT)*, vol. 23, no. 5, pp. 480–493, 2012.
5. A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors," in *International Conference on Local Computer Networks (LCN)*. IEEE, 2004.
6. E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch, and T. C. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Turin, Italy: IEEE, 14-19 April 2013, pp. 79–80.
7. X. Vilajosana, J. Llosa, I. Vilajosana, and J. Prieto-Blazquez, "Arp@: Remote Experiences with Real Embedded Systems," *Computer Applications in Engineering Education*, vol. 22, no. 4, pp. 639–648, 2014.
8. D. Stanislowski, X. Vilajosana, Q. Wang, T. Watteyne, and K. S. Pister, "Adaptive Synchronization in IEEE802.15.4e Networks," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 795–802, 2014.

9. *IEEE802.15.4e-2012: IEEE Standard for Local and Metropolitan Area Networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer*, IEEE Computer Society Std., Rev. IEEE Std 802.15.4e-2012, 16 April 2012.
10. T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, IETF Std. RFC6550, March 2012.
11. Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, IETF Std. RFC7252, June 2014.
12. P. Thubert, T. Watteyne, M.-R. Palattella, X. Vilajosana, and Q. Wang, "IETF 6TSCH: Combining IPv6 Connectivity with Industrial Performance," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, July 2013, pp. 541–546.
13. P. Tuset-Peiro, F. Vazquez-Gallego, J. Alonso-Zarate, L. Alonso, and X. Vilajosana, "LPDQ: A Self-scheduled TDMA MAC Protocol for One-hop Dynamic Low-power Wireless Networks," *Pervasive and Mobile Computing*, vol. 20, no. 0, pp. 84 – 99, 2015.
14. F. J. Chraim, "Wireless Sensing Applications for Critical Industrial Environments," Ph.D. dissertation, University of California at Berkeley, 13 December 2014, uCB/EECS-2014-215.
15. M. L. Fairbairn, "Dependability of Wireless Sensor Networks," Ph.D. dissertation, University of York, September 2014.
16. V. Pimentel, "Estimating the Safety Function Response Time for Wireless Control Systems," Master's thesis, University of New Brunswick, March 2015.
17. T. Watteyne, I. Robles, and X. Vilajosana, "Low-power, Lossy Network Plugfest Demonstrates Running Internet of Things Code," *IETF Journal*, vol. 10, no. 2, pp. 18–20, November 2014.
18. L. Grieco, A. Rizzo, S. Colucci, S. Sicari, G. Piro, D. Di Paola, and G. Boggia, "IoT-aided Robotics Applications: Technological Implications, Target Domains and Open Issues," *Computer Communications*, vol. 64, pp. 32–47, December 2014.
19. A. Mathur and T. Newe, "Secure Wireless Sensor Network for Medical Applications," in *NUIG-UL Alliance Conference*, April 2015.
20. K. P. Weekly, "Applied Estimation of Mobile Environments," Ph.D. dissertation, University of California, Berkeley, 2014.