On Real-Time Requirements in Constrained Wireless Networks for Mobile Health

Oliver Hahm
INRIA, France
oliver.hahm@inria.fr

Stefan Pfeiffer
Computer Systems and Telematics
Institute of Computer Science
Freie Universität Berlin, Germany
pfeiffer@inf.fu-berlin.de

Jochen Schiller
Computer Systems and Telematics
Institute of Computer Science
Freie Universität Berlin, Germany
schiller@inf.fu-berlin.de

ABSTRACT

This paper analyzes the requirements of mobile health applications concerning real-time criteria and describes the current state of real-time capabilities on constrained devices and low-power networks. Based on this analysis we observe that for these applications real-time capabilities are not only required per system, but also for the entire distributed system. Furthermore, we describe which technologies are available for the network stack, the software platform, and the hardware in order to fulfill these requirements. From the requirements on the network stack, following a top-down approach, we derive hardware prerequisites. We then conduct measurements on typical IoT hardware and operating system. We conclude that it is feasible to fulfill the identified prerequisites.

Categories and Subject Descriptors

C.3 [**Special-purpose and Application-based systems**]: [real-time and embedded systems, microprocessor/microcomputer applications]; C.2.1 [**Computer-communication Networks**]: Network Architecture and Design—*wireless communication*

Keywords

Mobile Health; Measurements; Deterministic Networking

1. INTRODUCTION

When industry began to deploy computers in factories, power plants, vehicles, and hospitals to automate the control in these mixed critical systems, a bunch of new protocols and wired buses emerged. Their specifications were specially designed to enable communication within these systems. Some well-known examples for theses protocols and buses are CAN-bus, M-Bus, HART, and BACnet. Over the last two decades a trend has evolved to replace these specialized communication systems by more generic solutions based on Ethernet and IP. Since connectivity between various, often heterogeneous networks and Internet hosted services has become more and more important, the deployment of specialized gateways increases complexity and decreases efficiency. Nowadays a significant amount of deployed systems in building and industrial automation, car entertainment systems, and medical appliances are based on IP and Ethernet [1]. However, the use of IP over wireless networks is still a rare case. While specialized wireless technologies based on technologies like IEEE 802.15.4, Wireless HART, or DECT are already widely used, the applicability of IP, particular over multi-hop wireless networks is still considered to pose too many problems to the system to be used in critical systems. Nevertheless, the increasing requirements to interconnect various systems and attach them to the Internet, make us believe that IP driven systems will emerge also in the domain of wireless networks with mixed criticality. These systems will also have a need for a certain degree of multi-hop communication, since one cannot expect to have sufficiently good wireless link to every node, even in a small space such as a hospital room [2]. Hence, even though the maximum number of hops most likely will not be very high (i.e. could be as low as 2), the network must be able to perform self-configuration and self-healing features over multi-hop communication.

The use cases for mobile health appliances are good examples for such systems. A typical scenario for a mobile health application is depicted in figure 1. Here, the patient is monitored by various devices. These devices, like normal wireless sensor nodes, are equipped with an highly energy-efficient micro-controller, a low-power radio transceiver, and one or more sensors that can, for example, measure the blood pressure or the heart rate. Characteristic properties for these constrained nodes are defined in RFC 7228 [3]. Using their radio transceivers, they can build a wireless body area network (WBAN) and may connect eventually to other close-by systems like smartphones or home gateway to access services in the Internet.

In this paper we will shed a light on the particular requirements of mobile health applications on the network. Section 2 will present these requirements in detail. The following section 3 will then present the state of the art concerning communication protocols, software platforms, and

MobileHealth'14, August 11, 2014, Philadelphia, PA, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2983-5/14/08i...$15.00.
http://dx.doi.org/10.1145/2633651.2637475 .

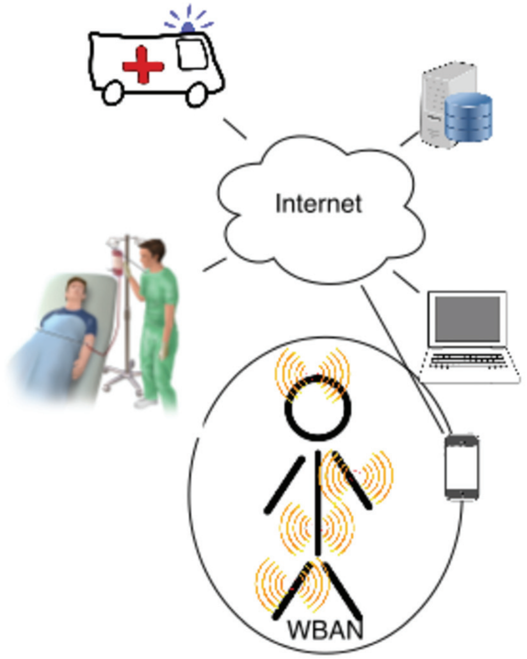

Figure 1: Typical mobile health scenario with various health monitoring sensors connected to each other in a WBAN. More powerful mobile devices like smartphones and laptops or fixed devices within a hospital room can serve as border routers to connect the WBAN to the Internet and access online medical records or call the ambulance in case of an emergency.

hardware. Finally, section 4 presents the results of some measurements on realistic systems concerning the described requirements.

2. REQUIREMENTS OF MOBILE HEALTH APPLICATIONS

The term ”real-time” can be found in almost every publication on mobile health applications. However, throughout literature it is often used with differing notions. In many cases the term refers to the capability of a system to perform live monitoring or online processing of data. In contrary, this paper will analyze the real-time requirements of mobile health applications in a stricter definition of the term as denoted by Hou and Kumar [4]. According to this definition ”a real-time system is required to complete its work and deliver its services on a timely basis”. The following paragraphs will describe typical challenges of mobile health applications and derive subsequently the requirements on the real-time capabilities of the system. Furthermore, it is important to consider the real-time properties on a single machine and for the entire distributed system. As we will see, the latter one is dependent on the first one.

We distinguish the following main requirements for mobile health applications: (1) accuracy, (2) reliability, (3) robustness, (4) timeliness, and (5) wearability.

2.1 Accuracy

Accurate measurements are obviously a strong requirement for health monitoring systems. This applies for the precision of the sensor data as well as for the exact time-stamping of the measured values. While the first demand can be mostly tackled during the process of hardware design and applying appropriate filter techniques, the second one is challenging for the system software itself. Exact time-stamping require deterministic system behavior to minimize the jitter of the time-stamps as much as possible. It also requires clock synchronization techniques in order to provide comparable time-stamps.

2.2 Reliability

Providing accurate measurements is not sufficient, but reliable transmission of these values is also mandatory. The problem of reliable communication can be tackled at various layers. Locally performed feature extraction may serve to reduce heavy demands on the communication. Using predictable and deterministic MAC layer protocols as described in section 3 can minimize the probability of data loss.

Additionally, several of the envisioned mobile health applications have the need to work together in a synchronized manner. Especially for these systems it is of importance to minimize jitter at all layers. This means that already the exact moment of the measurement must vary as little as possible and that network communication up to the delivery to the application itself should add ideally no temporal variance.

2.3 Robustness

The heterogeneity of mobile health systems and their deployment fields demands techniques to harden the network against interferences. TDMA based MAC protocols help to avoid collisions on the medium and, thus, minimize internal interference. Channel assignment algorithms and FDMA MAC protocols can serve to make better use of the available frequency bandwidth in order to mitigate internal as well as external interferences [5].

2.4 Timeliness

Particular for live critical systems it is of prime importance to transmit information timely. Therefor, it is crucial to schedule the transmissions in a way that potential deadlines of the application can be fulfilled. Provisioning such a schedule, and thereby guaranteeing certain deadlines between two endpoints, increases also the possibility to predict the overall throughput of every single node in the network. In order to establish such a schedule a tight cooperation between the MAC layer and upper layers is required.

2.5 Wearability

There are two main problems with traditional health monitoring devices, such as Holter monitors [6]: (1) Since the data is processed offline, they are not able to perform continual online monitoring and early detection of medical disorders. (2) They limit the patient’s activity and level of comfort, which influences the measured results. In order to avoid these problems, the next generation of health monitoring device has to be as small as possible and work autonomously. The technical advances of miniaturization and integration, wireless networking, and energy efficiency are enablers for these new devices. However, the size of the battery is still the limiting factor for the size and weight of sensor devices. Thus, in order to enable long-term monitoring of a patient's medical situation, the mobile health devices have to make a high use of the hardware's power saving capabilities. Fine granular measurements show that maximum energy efficiency can only be achieved by sending the device, i.e. the micro-controller and the radio chip, as much as possible to power-down modes [7]. This in turn makes the design of network stacks based on time-slotted medium access very attractive for health monitoring devices.

3. STATE OF THE ART

This section will present the state of the art technologies that are available to implement mobile health applications. From the requirements as identified in section 2 we derive requirements for the software stack which then in turn defines the requirements for the hardware platform.

3.1 Network Stacks

Taking into account the described requirements for mobile health applications, researchers, engineers, and protocol designers have developed several techniques to tackle the challenges posed by a wireless multi-hop network. In this paper we will focus on two key aspects of this problem space: the particular task of the media access control (MAC) and the design of a full network stack able to give the required guarantees.

3.1.1 MAC protocols

In order to fulfill the described requirements, the MAC protocol has a pivotal role. CSMA, TDMA, and FDMA based channel access methods are typically used in the area of low-power and lossy networks. While CSMA as it used for traditional IEEE 802.15.4 networks has the lowest requirements on the hardware and is comparable easy to implement, it can only give very little guarantees and is therefore not suitable for most mobile health applications [8]. On the contrary, TDMA and FDMA make additional demands on the hardware and software, such as guaranteed transceiver switching times or clock synchronization, but are more robust, reliable, and able to provide a deterministic behavior. Typical protocols in this area are T-MAC [9] and Bluetooth Low Energy [10]. In 2012 IEEE published IEEE 802.15.4e standard which comprises an amendment to the original MAC protocol of IEEE 802.15.4, introducing a Timeslotted Channel Hopping (TSCH) mode [11]. This mode describes a combination of a TDMA and FDMA channel access method and aims to enable high reliability and robustness at little energy. Nodes in a IEEE 802.15.4e network maintain a transmission schedule. This schedule consists of timeslots that provide a unit of bandwidth, allocated for the communication to its neighbors. In order to enable predictable transmission patterns, the allocation of these slots is configurable. For the rest of this paper, we will therefore focus on IEEE 802.15.4e.

3.1.2 Complete Network Stacks

As described in subsection 2.4 a tight interaction between upper layers and the MAC protocol is mandatory to fulfill the real-time requirements of the system. On the one hand, there is a need for supportive upper layer services like clock synchronization and scheduling algorithms, on the other hand, routing and transport layer services also have to be aware of the particular requirements and properties of these MAC layer protocols. While recent research has proposed some approaches to time synchronization and TDMA scheduling algorithms, holistic approaches are rarely examined [12, 13].

Recently, following the paradigm of the Internet of Things, a trend evolved to connect all kind of machine-to-machine systems, wired and wireless directly to the Internet [14]. The deployment of standard IP suite protocols in these networks is essential for this Internet connectivity. The capability of communicating end-to-end over IP is the key for a new generation of mobile health applications, too. It allows to exchange data not only between the health monitoring devices in a WBAN, but seamlessly connect to remote services in the Internet. By this means health monitoring devices can perform live updates of medical records or send an alarm in life threatening situations.

However, the combination of both, deterministic MAC layer protocols as proposed by IEEE 802.15.4e and the deployment of an IP-based network stack, require some further considerations. Since IEEE 802.15.4e only defines the link-layer mechanisms, but not how the transmission schedule is built and maintained to match the traffic requirements, therefore, an upper layer service is needed. The IETF working group *6tisch* is currently chartered to define such an operation sublayer and make the IP protocol suite capable to run on top of IEEE 802.15.4e. One of the first implementations of the entities proposed by this working group is the *OpenWSN* network stack [15].

OpenWSN by default uses timeslots with duration of 15 ms. The proposed timing constants for this configuration that are based on measurements range from tens of microseconds up to a few milliseconds. The guard time, for example, which determines how long the transceiver has to be active, and, therefore, should kept as small as possible, is configured to 1.3 ms. Hence, the software must guarantee a worst-case execution time for handling an external event significantly below this value, in order to process an incoming packet in time.

3.2 Software Platforms

As seen in the preceding sub-section, special care has to be taken, designing the software platform for mobile health applications. The first design decision for the software architect is whether to build the system from scratch or choose an existing operating system (OS). If the latter option is chosen, one has to select between several OS designs and architectures. The OS design has a heavy impact on crucial properties of the system like real-time capabilities, (energy) efficiency, reliability, and flexibility. In particular, the choice of the OS architecture defines if and which real-time requirements can be fulfilled.

Traditionally, real-time critical systems were implemented from scratch by developing a software that is especially tailored to match exactly the needs of the application scenario. Since these systems are obviously smaller and less complex than systems based on general purpose OS', they are easier to verify and, therefore, make certification less expensive. However, it is also evident that this approach is neither very scalable nor flexible. Even the deployment of software updates can become a difficult and time-consuming task. Thus, developing and maintaining this type of specialized software stack increases the costs. Nevertheless, the design and implementation of a more flexible OS is a challenging task and requires the systematic enforcement of programming concepts.

The typical design of OS' for constrained-node networks (as defined in RFC 7228 [3]) is based on an event-based programming paradigm like in Contiki or TinyOS [16]. For time-critical systems usually dedicated real-time OS' (RTOS) like QNX are deployed [17]. Other RTOS' like FreeRTOS and RIOT have been designed, in order to match the requirements of a low memory footprint and high energy efficiency as they are typical for constrained-node networks [18,19]. The latter ones match the requirements of memory and energy constrained nodes by providing a small memory footprint and using a tickless scheduler in order to spend as much time as possible in deep sleep mode. In order to guarantee real-time capabilities they are designed with the following properties: (1) a priority-based, preemptive scheduler, (2) every kernel functionality has deterministic runtime, i.e. for example, no standard `malloc()` within the kernel, and (3) minimalistic interrupt handlers. For the evaluation we used RIOT, since it provides multiple network stack implementations and several useful system libraries already built-in, where as FreeRTOS itself can be considered rather as a *thread library* than a full-fledged OS.

3.3 Hardware

At this point of the analysis we are ready to define the requirements and constraints concerning the hardware platforms that can be used to develop mobile health appliances. These definitions can be divided into two categories concerning the following aspects: (1) interrupt latency and (2) clock properties and wake-up times.

3.3.1 Interrupt Latencies

Interrupt latencies are central to mobile health appliances regarding two features: (1) they must provide a minimal delay between the occurrence of the event and handling of the interrupt and (2) they must guarantee a small jitter for this delay. Both features are important for the measurement on the monitoring device itself, but are also of importance to the MAC layer, in order to keep in sync with the other nodes. The guard time in TSCH mode of IEEE 802.15.4e, for instance, is dependent on this latency.

3.3.2 Clock Properties and Wake-Up Times

The clock frequency defines the resolution of the timers and, therefore, the maximum accuracy of time-stamps. Another important characteristic is its stability, which defines the clock drift. This property is usually influenced by temperature, voltage, and age of the deployed crystal. Good values for the usage of IEEE 802.15.4e are timer clocked by at least a 32 kHz crystal and a rather stable drift of maximum 20 to 30 ppm. It is also crucial how fast the system is able to wake up from sleep modes. For IEEE 802.15.4e the timer should be able to wake up the micro-controller in less than 100 microseconds from any deep sleep mode with RAM retention.

4. EXPERIMENTAL RESULTS

In order to analyze if the requirements on the hardware and software, we derived in section 3 can be fulfilled using IoT hardware and a typical IoT OS, we performed some measurements. For the OS, we have chosen RIOT because (i) it has real-time properties and (ii) it supports OpenWSN and its IEEE 802.15.4e implementation. The OpenWSN network stack can be linked using a BSD like package system. For the hardware, we have chosen a ScatterWeb MSB-A2, which is based on an ARM7 micro-controller with 98 kB RAM and 512 kB ROM that speeds up to 72 MHz [20]. This type of sensor board can be considered as a typical design for an IoT device, in terms of the micro-controller architecture and the available memory. The board has been developed and used in research and teaching over the last decade at Freie Universität Berlin. It is equipped with the widely used TI CC1100 radio transceiver at 868 MHz and is also deployed in the DES-Testbed [21].

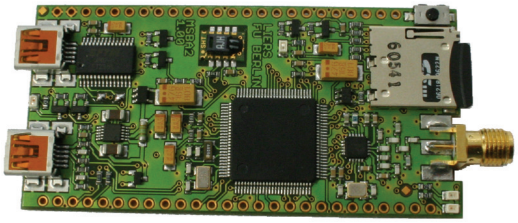

Figure 2: An MSB-A2 node.

For the measurements we used two different setups: (1) In setup I we connected the GPIO pins G_{OUT1} and G_{OUT2} of the sensor boards (configured as output) with the probes of an oscilloscope. (2) In setup II we additionally connected the GPIO pins G_{TRIG} of another sensor node to the GPIO pins G_{IN} of the first one, configured as input this time.

The properties that need to be measured are determined both by the hardware and the OS. In section 4.1 we report on measurements of typical IoT hardware, while in section 4.2, we present measurements of typical IoT OS performance.

4.1 Measurement of the Hardware Properties

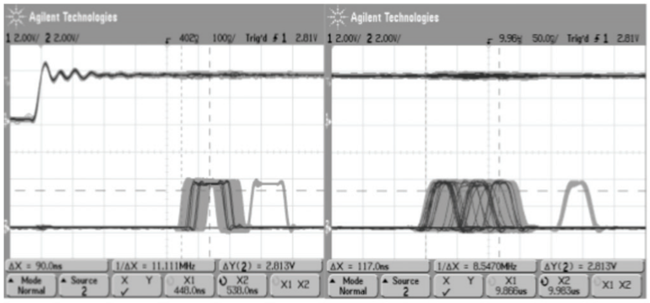

Figure 3: Screenshot from the oscilloscope measuring the delay and jitter of the IRQ and FIQ on the MSB-A2. Channel 1 (the upper one) triggers on the output of G_{TRIG}, channel 2 is raised when the ISR is called. The left picture shows the results for the FIQ, the right one for the normal IRQ.

With the measurements presented in this subsection we evaluated what can be achieved in practice on the given hardware concerning real-time constraints. The goal of these measurements was to check if the analyzed micro-controller can provide the necessary limits for delay and variance of interrupt handling. For the first measurement we used setup I and modified the MSB-A2, so that the *GDO2* pin of the

CC1100 radio chip is connected with the capture register of one of the micro-controller's timers. Then we generated square-wave signal on the CC1100 with a frequency of 1.08 MHz. The timer counted up and captured on every rising flag of the CC1100's *GDO2* pin. Finally, the captured values were printed over the USART. As a result we observed a variation of *13.9 ns* which is magnitudes smaller than what is needed for IEEE 802.15.4e time-stamping. Actually, this interval matches exactly the duration of one cycle on micro-controller at 72 MHz and is, therefore, the best achievable value for the jitter.

In the second measurement we examined the delay and jitter between the occurrence of an interrupt causing event and the point of time when the ARM interrupt service routine (ISR) gets called. We performed the experiment with the normal interrupt request (IRQ) and so called fast interrupts (FIQ) on the ARM7. Therefore, we connected in setup I G_{OUT1} to the first probe of the oscilloscope and G_{OUT2} to the other probe. We then measured the time between the two edges of the output signals to determine the interval. For the FIQ we measured a delay of about 450 ns and approximately 9.9 µs for the normal IRQ. In both cases we observed a randomly distributed jitter of about 90 µs and a peridiocal jitter of 230 ns. Again, neither this amount of delay nor the jitter are problematic for IEEE 802.15.4e time-stamping.

Thus, we can derive that the ARM7 which powers the MSB-A2 is in general usable for IEEE 802.15.4e. Since this node is comparable to most IoT hardware, we can assume that the implementation of a deterministic network stack on these devices is feasible.

4.2 Measurement of the Software Properties

With these measurements we evaluated if the required timings can be achieved, using a typical IoT OS, in worst case scenarios. This is important to validate if the system can still work correctly, i.e. fulfill the necessary deadlines, under all circumstances. Therefore, we implemented our test in RIOT and put the system under test under full load, by creating a thread A that accesses a peripheral device (in our tests, the USART) constantly in a while loop. A was created with a priority P_A whereas the main thread has priority P_M with $P_A > P_M$, so that A would run without cease, if no other event would occur. Then we created a second thread B with priority $P_B > P_A$, but put it immediately into sleep state by calling `thread_sleep()`. Based on this setup we ran two different tests: In the first one, using setup I, we registered an interrupt handler for one of the hardware timers, and in the second one, using setup II, we registered an interrupt handler for an external GPIO interrupt G_IN. In both cases the interrupt handler was configured to wake up B by calling `thread_wakeup(pid_of_B)`. B will toggle the output GPIO pin connected to the second probe of the oscilloscope as soon as it wakes up again.

For the first test we measured a delay between the ARM ISR and B being awake again of 14.9 µs and jitter in magnitudes of nanoseconds. Taking into account the values we measured in the test 4.1, the maxium delay is still below 15 µs which is tolerable for time-stamping in IEEE 802.15.4e.

The second test revealed a delay between the occurrence of the IRQ and B being awake again of 20.8 µs. Again, this value is clearly below the values proposed in OpenWSN as guard times. The higher delay for the GPIO interrupt is caused by the design of RIOT's GPIO interrupt handler, which multiplexes all available GPIO interrupts in one handler function.

In order to compare these results, we conducted the same measurements without A. The results were almost identical (± one CPU cycle). Therefore, it can be deduced that the response time for critical events of RIOT and the whole system is not influenced by the load of the system, if the priorities of the threads are set correctly.

Hence, we can conclude that both, the evaluated standard sensor board and the OS, RIOT, are capable to guarantee the required response times that we derived in section 3. These results make us optimistic that the deployment of IP over wireless networks are feasible in scenarios with hard real-time requirements such as for mobile health applications.

5. CONCLUSION

In this paper we considered the requirements of mobile health applications regarding real-time properties. We described the features the network stack has to provide in order to fulfill these demands. From that, we derived the required response time capabilities of the software stack which in turn depends on timings provided by the hardware. We then measured the achievable performance with respect to basic real-time properties, using standard IoT hardware and OS. We conclude that it is indeed possible to fulfill the identified requirements with this hardware and software.

Acknowledgment

We would like to thank Emmanuel Baccelli for his helpful comments and discussions on this topic.

6. REFERENCES

[1] P. Neumann, "Communication in industrial automation—what is going on?" *Control Engineering Practice*, vol. 15, no. 11, pp. 1332–1347, 2007.

[2] D. Kotz, C. Newport, and C. Elliott, "The mistaken axioms of wireless-network research," 2003.

[3] C. Bormann, M. Ersue, and A. Keranen, "Terminology for constrained node networks," 2013.

[4] K. P. Hou, I-Hong, *Packets with Deadlines*. Morgan & Clayland Publishers, 2013.

[5] F. Juraschek, S. Seif, and M. Güneş, "Distributed Channel Assignment in Large-Scale Wireless Mesh Networks: A Performance Analysis," in *IEEE International Conference on Communications (ICC)*, June 2013.

[6] A. Milenković, C. Otto, and E. Jovanov, "Wireless sensor networks for personal health monitoring: Issues and an implementation," *Computer communications*, vol. 29, no. 13, pp. 2521–2533, 2006.

[7] O. Hahm and S. Adler, "Profiling energy consumption of wireless sensor nodes with almost zero effort," in *Communications (ICC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 6273–6277.

[8] IEEE standard for Information Technology, "IEEE std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks," June 2011.

[9] T. Van Dam and K. Langendoen, “An adaptive energy-efficient MAC protocol for wireless sensor networks,” in *Proceedings of the 1st international conference on Embedded networked sensor systems.* ACM, 2003, pp. 171–180.

[10] B. S. I. Group, “Bluetooth Core Specification Version 4.1,” IEEE, December 2013.

[11] IEEE standard for Information Technology, “IEEE std. 802.15.4e, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer,” April 2012.

[12] K. Römer, P. Blum, and L. Meier, *Time Synchronization and Calibration in Wireless Sensor Networks.* Wiley and Sons, October 2005.

[13] S. C. Ergen and P. Varaiya, “TDMA scheduling algorithms for wireless sensor networks,” *Wireless Networks*, vol. 16, no. 4, pp. 985–997, 2010.

[14] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, Eds., *Vision and challenges for realising the Internet of Things.* Cluster of European Research Projects on the Internet of Things, European Commision, 2010.

[15] 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*, vol. 23, no. 5, pp. 480–493, 2012.

[16] M. O. Farooq and T. Kunz, “Operating systems for wireless sensor networks: A survey,” *Sensors Journal*, 2011.

[17] D. C. Sastry and M. Demirci, “The QNX operating system,” *Computer*, vol. 28, no. 11, pp. 75–77, 1995.

[18] R. Barry, “FreeRTOS,” June 2014. [Online]. Available: http://www.freertos.org/

[19] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. C. Schmidt, “RIOT OS: Towards an OS for the Internet of Things,” in *Proc. of the 32nd IEEE INFOCOM. Poster.* Piscataway, NJ, USA: IEEE Press, 2013.

[20] M. Baar, H. Will, B. Blywis, A. Liers, G. Wittenburg, and J. Schiller, “The ScatterWeb MSB-A2 Platform for Wireless Sensor Networks,” Freie Universität Berlin, Tech. Rep., 2008.

[21] M. Günes, B. Blywis, M. Frey, O. Hahm, F. Juraschek, P. Kumar, Q. Mushtaq, and K. Schleiser, “DES-Testbed,” http://www.des-testbed.net, 2008-2014, homepage of the DES-Testbed. [Online]. Available: http://www.des-testbed.net