

Article

Wireless Sensor Network Operating System Design Rules Based on Real-World Deployment Survey

Girts Strazdins ^{1,2,*}, Atis Elsts ^{1,2}, Krisjanis Nesenbergs ^{1,2} and Leo Selavo ^{1,2}

¹ Institute of Electronics and Computer Science, 14 Dzerbenes Street, Riga, LV 1006, Latvia;
E-Mails: atis.elsts@edi.lv (A.E.); krisjanis.nesenbergs@edi.lv (K.N.); leo.selavo@edi.lv (L.S.)

² Faculty of Computing, University of Latvia, 19 Raina blvd, Riga, LV 1586, Latvia

* Author to whom correspondence should be addressed; E-Mail: girts.strazdins@edi.lv;
Tel.: +371-6755-8224; Fax: +371-6755-5337.

Received: 1 June 2013; in revised form: 13 July 2013 / Accepted: 24 July 2013 /

Published: 16 August 2013

Abstract: Wireless sensor networks (WSNs) have been a widely researched field since the beginning of the 21st century. The field is already maturing, and TinyOS has established itself as the *de facto* standard WSN Operating System (OS). However, the WSN researcher community is still active in building more flexible, efficient and user-friendly WSN operating systems. Often, WSN OS design is based either on practical requirements of a particular research project or research group's needs or on theoretical assumptions spread in the WSN community. The goal of this paper is to propose WSN OS design rules that are based on a thorough survey of 40 WSN deployments. The survey unveils trends of WSN applications and provides empirical substantiation to support widely usable and flexible WSN operating system design.

Keywords: wireless sensor networks; deployment; survey; operating system; design rules

1. Introduction

Wireless sensor networks are a relatively new field in computer science and engineering. Although the first systems that could be called WSNs were used already in 1951, during the Cold War [1], the real WSN revolution started in the beginning of the 21st century, with the rapid advancement of micro-electro-mechanical systems (MEMS). New hardware platforms [2,3], operating systems [4,5],

middleware [6,7], networking [8], time synchronization [9], localization [10] and other protocols have been proposed by the research community. The gathered knowledge has been used in numerous deployments [11,12]. TinyOS [4] has been the *de facto* standard operating system in the community since 2002. However, as the survey will reveal, customized platforms and operating systems are often used, emphasizing the still actual WSN user need for a flexible and easily usable OS.

The goal of this paper is to summarize WSN deployment surveys and analyze the collected data in the OS context, clarifying typical deployment parameters that are important in WSN OS design.

2. Methodology

Research papers presenting deployments are selected based on multiple criteria:

- The years 2002 up to 2011 have been reviewed uniformly, without emphasis on any particular year. Deployments before the year 2002 are not considered, as early sensor network research projects used custom hardware, differing from modern embedded systems significantly.
- Articles have been searched using the Association for Computing Machinery (ACM) Digital Library (<http://dl.acm.org/>), the Institute of Electrical and Electronics Engineers (IEEE) Xplore Digital Library (<http://ieeexplore.ieee.org/>), Elsevier ScienceDirect and SpringerLink databases. Several articles have been found as external references from the aforementioned databases.
- Deployments are selected to cover a wide WSN application range, including environmental monitoring, animal monitoring, human-centric applications, infrastructure monitoring, smart buildings and military applications.

WSN deployment surveys can be found in the literature [13–18]. This survey focuses on more thorough and detailed review regarding the aspects important for WSN OS design. This survey also contains deployments in the prototyping phase, because of two reasons. First, rapid prototyping and experimentation is a significant part of sensor network application development. Second, many of the research projects develop a prototype, and stable deployments are created later as commercial products, without publishing technical details in academic conferences and journals. Therefore software tools must support experimentation and prototyping of sensor networks, and the requirements of these development phases must be taken into account.

Multiple parameters are analyzed for each of the considered WSN deployments. For presentation simplification, these parameters are grouped, and each group is presented as a separate subsection.

For each deployment, the best possible parameter extraction was performed. Part of information was explicitly stated in the analyzed papers and web pages, and part of it was acquired by making a rational guess or approximation. Such approximated values are marked with a question mark right after the approximated value.

3. Survey Results

The following subsections describe parameter values extracted in the process of deployment article analysis. General deployment attributes are shown in Table 1. Each deployment has a *codename*

assigned. This will be used to identify each article in the following tables. Design rules are listed in the text right after conclusions substantiating the rule.

The extracted design rules should be considered as WSN deployment trends that suggest particular design choices to OS architects. There is no strict evidence that any particular deployment trend must be implemented in an operating system at all costs. These design rules sketch likely choices of WSN users that should be considered.

Table 1. Deployments: general information.

Nr	Codename	Year	Title	Class	Description
1	Habitats [11]	2002	Wireless Sensor Networks for Habitat Monitoring	Habitat and weather monitoring	One of the first sensor network deployments, designed for bird nest monitoring on a remote island
2	Minefield [12]	2003	Collaborative Networking Requirements for Unattended Ground Sensor Systems	Opposing force investigation	Unattended ground sensor system for self healing minefield application
3	Battlefield [19]	2004	Energy-Efficient Surveillance System Using Wireless Sensor Networks	Battlefield surveillance	System for tracking of the position of moving targets in an energy-efficient and stealthy manner
4	Line in the sand [20]	2004	A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking	Battlefield surveillance	System for intrusion detection, target classification and tracking
5	Counter-sniper [21]	2004	Sensor Network-Based Countersniper System	Opposing force investigation	An <i>ad hoc</i> wireless sensor network-based system that detects and accurately locates shooters, even in urban environments.
6	Electro-shepherd [22]	2004	Electronic Shepherd—A Low-Cost, Low-Bandwidth, Wireless Network System	Domestic animal monitoring and control	Experiments with sheep GPS and sensor tracking
7	Virtual fences [23]	2004	Virtual Fences for Controlling Cows	Domestic animal monitoring and control	Experiments with virtual fence for domestic animal control
8	Oil tanker [24]	2005	Design and Deployment of Industrial Sensor Networks: Experiences from a Semiconductor Plant and the North Sea	Industrial equipment monitoring and control	Sensor network for industrial machinery monitoring, using Intel motes with Bluetooth and high-frequency sampling
9	Enemy vehicles [25]	2005	Design and Implementation of a Sensor Network System for Vehicle Tracking and Autonomous Interception	Opposing force investigation	A networked system of distributed sensor nodes that detects an evader and aids a pursuer in capturing the evader
10	Trove game [26]	2005	Trove: A Physical Game Running on an <i>Ad hoc</i> Wireless Sensor Network	Child education and sensor games	Physical multiplayer real-time game, using collaborative sensor nodes
11	Elder Radio-Frequency Identification (RFID) [27]	2005	A Prototype on RFID and Sensor Networks for Elder Healthcare: Progress Report	Medication intake accounting	In-home elder healthcare system integrating sensor networks and RFID technologies for medication intake monitoring
12	Murphy potatoes [28]	2006	Murphy Loves Potatoes: Experiences from a Pilot Sensor Network Deployment in Precision Agriculture	Precision agriculture	A rather unsuccessful sensor network pilot deployment for precision agriculture, demonstrating valuable lessons learned
13	Firewxnet [29]	2006	FireWxNet: A Multi-Tiered Portable Wireless System for Monitoring Weather Conditions in Wildland Fire Environments	Forest fire detection	A multi-tier WSN for safe and easy monitoring of fire and weather conditions over a wide range of locations and elevations within forest fires
14	AlarmNet [30]	2006	ALARM-NET: Wireless Sensor Networks for Assisted-Living and Residential Monitoring	Human health telemonitoring	Wireless sensor network for assisted-living and residential monitoring, integrating environmental and physiological sensors and providing end-to-end secure communication and sensitive medical data protection
15	Ecuador Volcano [31]	2006	Fidelity and Yield in a Volcano Monitoring Sensor Network	Volcano monitoring	Sensor network for volcano seismic activity monitoring, using high frequency sampling and distributed event detection
16	Pet game [32]	2006	Wireless Sensor Network-Based Mobile Pet Game	Child education and sensor games	Augmenting mobile pet game with physical sensing capabilities: sensor nodes act as eyes, ears and skin

Table 1. *Cont.*

Nr	Codename	Year	Title	Class	Description
17	Plug [33]	2007	A Platform for Ubiquitous Sensor Deployment in Occupational and Domestic Environments	Smart energy usage	Wireless sensor network for human activity logging in offices; sensor nodes implemented as power strips
18	B-Live [34]	2007	B-Live—A Home Automation System for Disabled and Elderly People	Home/office automation	Home automation for disabled and elderly people integrating heterogeneous wired and wireless sensor and actuator modules
19	Biomotion [35]	2007	A Compact, High-Speed, Wearable Sensor Network for Biomotion Capture and Interactive Media	Smart user interfaces and art	Wireless sensor platform designed for processing multipoint human motion with low latency and high resolutions. Example applications: interactive dance, where movements of multiple dancers are translated into real-time audio or video
20	AID-N [36]	2007	The Advanced Health and Disaster Aid Network: A Light-Weight Wireless Medical System for Triage	Human health telemonitoring	Lightweight medical systems to help emergency service providers in mass casualty incidents
21	Firefighting [37]	2007	A Wireless Sensor Network and Incident Command Interface for Urban Firefighting	Human-centric applications	Wireless sensor network and incident command interface for firefighting and emergency response, especially in large and complex buildings. During a fire accident, fire spread is tracked, and firefighter position and health status are monitored.
22	Rehabil [38]	2007	Ubiquitous Rehabilitation Center: An Implementation of a Wireless Sensor Network-Based Rehabilitation Management System	Human indoor tracking	Zigbee sensor network-based ubiquitous rehabilitation center for patient and rehabilitation machine monitoring
23	CargoNet [39]	2007	CargoNet: A Low-Cost Micropower Sensor Node Exploiting Quasi-Passive Wake Up for Adaptive Asynchronous Monitoring of Exceptional Events	Good and daily object tracking	System of low-cost, micropower active sensor tags for environmental monitoring at the crate and case level for supply-chain management and asset security
24	Fence monitor [40]	2007	Fence Monitoring—Experimental Evaluation of a Use Case for Wireless Sensor Networks	Security systems	Sensor nodes attached to a fence for collaborative intrusion detection
25	BikeNet [41]	2007	The BikeNet Mobile Sensing System for Cyclist Experience Mapping	City environment monitoring	Extensible mobile sensing system for cyclist experience (personal, bicycle and environmental sensing) mapping, leveraging opportunistic networking principles
26	BriMon [42]	2008	BriMon: A Sensor Network System for Railway Bridge Monitoring	Bridge monitoring	Delay tolerant network for bridge vibration monitoring using accelerometers. Gateway mote collects data and forwards opportunistically to a mobile base station attached to a train passing by.
27	IP net [43]	2008	Experiences from Two Sensor Network Deployments—Self-Monitoring and Self-Configuration Keys to Success	Battlefield surveillance	Indoor and outdoor surveillance network for detecting troop movement
28	Smart home [44]	2008	The Design and Implementation of Smart Sensor-Based Home Networks	Home/office automation	Wireless sensor network deployed in a miniature model house, which controls different household equipment: window curtains, gas valves, electric outlets, TV, refrigerator and door locks
29	SVATS [45]	2008	SVATS: A Sensor-Network-Based Vehicle Anti-Theft System	Anti-theft systems	Low cost, reliable sensor-network based, distributed vehicle anti-theft system with low false-alarm rate
30	Hitchhiker [46]	2008	The Hitchhikers Guide to Successful Wireless Sensor Network Deployments	Flood and glacier detection	Multiple real-world sensor network deployments performed, including glacier detection; experience and suggestions reported.
31	Daily morning [47]	2008	Detection of Early Morning Daily Activities with Static Home and Wearable Wireless Sensors	Daily activity recognition	Flexible, cost-effective, wireless in-home activity monitoring system integrating static and mobile body sensors for assisting patients with cognitive impairments

Table 1. *Cont.*

Nr	Codename	Year	Title	Class	Description
32	Heritage [48]	2009	Monitoring Heritage Buildings with Wireless Sensor Networks: The Torre Aquila Deployment	Heritage building and site monitoring	Three different motes (sensing temperature, vibrations and deformation) deployed in a historical tower to monitor its health and identify potential damage risks
33	AC meter [49]	2009	Design and Implementation of a High-Fidelity AC Metering Network	Smart energy usage	AC outlet power consumption measurement devices, which are powered from the same AC line, but communicate wirelessly to IPv6 router
34	Coal mine [50]	2009	Underground Coal Mine Monitoring with Wireless Sensor Networks	Coal mine monitoring	Self-adaptive coal mine wireless sensor network (WSN) system for rapid detection of structure variations caused by underground collapses
35	ITS [51]	2009	Wireless Sensor Networks for Intelligent Transportation Systems	Vehicle tracking and traffic monitoring	Traffic monitoring system implemented through WSN technology within the SAFESPOT Project
36	Underwater [52]	2010	Adaptive Decentralized Control of Underwater Sensor Networks for Modeling Underwater Phenomena	Underwater networks	Measurement of dynamics of underwater bodies and their impact in the global environment, using sensor networks with nodes adapting their depth dynamically
37	PipeProbe [53]	2010	PipeProbe: A Mobile Sensor Droplet for Mapping Hidden Pipeline	Power line and water pipe monitoring	Mobile sensor system for determining the spatial topology of hidden water pipelines behind walls
38	Badgers [54]	2010	Evolution and Sustainability of a Wildlife Monitoring Sensor Network	Wild animal monitoring	Badger monitoring in a forest
39	Helens volcano [55]	2011	Real-World Sensor Network for Long-Term Volcano Monitoring: Design and Findings	Volcano monitoring	Robust and fault-tolerant WSN for active volcano monitoring
40	Tunnels [56]	2011	Is There Light at the Ends of the Tunnel? Wireless Sensor Networks for Adaptive Lighting in Road Tunnels	Tunnel monitoring	Closed loop wireless sensor and actuator system for adaptive lighting control in operational tunnels

3.1. Deployment State and Attributes

Table 2 describes the deployment state and used sensor node (mote) characteristics. SVATS, sensor-network-based vehicle anti-theft system.

Table 2. Deployments: deployment state and attributes.

Nr	Codename	Deployment state	Mote count	Heterog. motes	Base stations	Base station hardware
1	Habitats	pilot	32	n	1	Mote + PC with satellite link to Internet
2	Minefield	pilot	20	n	0	All motes capable of connecting to a PC via Ethernet
3	Battlefield	prototype	70	y (soft, by role)	1	Mote + PC
4	Line in the sand	pilot	90	n	1	Root connects to long-range radio relay
5	Counter-sniper	prototype	56	n	1	Mote + PC
6	Electro-shepherd	pilot	180	y	1+	Mobile mote
7	Virtual fences	prototype	8	n	1	Laptop
8	Oil tanker	pilot	26	n	4	Stargate gateway + Intel mote, wall powered.

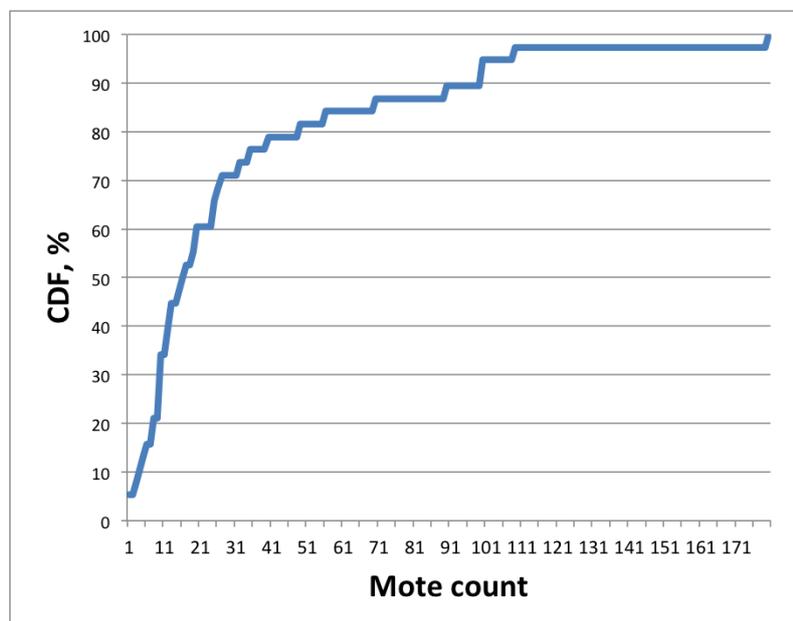
Table 2. Cont.

Nr	Codename	Deployment state	Mote count	Heterog. motes	Base stations	Base station hardware
9	Enemy vehicles	pilot	100	y	1	Mobile power motes - laptop on wheels
10	Trove game	pilot	10	n	1	Mote + PC
11	Elder RFID	prototype	3	n	1	Mote + PC
12	Murphy potatoes	pilot	109	n	1	Stargate gateway + Tnode, solar panel
13	Firewxnet	pilot	13	n	1 Base Station	Gateway: Soekris net4801 with Gentoo Linux and Trango Access5830 long-range (BS) + 5 10 Mbps wireless; BS: PC with satellite link 512/128Kbps
14	AlarmNet	prototype	15	y	varies	Stargate gateway with MicaZ, wall powered.
15	Ecuador Volcano	pilot	19	y	1	Mote + PC
16	Pet game	prototype	?	n	1+	Mote + MIB510 board + PC
17	Plug	pilot	35	n	1	Mote + PC
18	B-Live	pilot	10+	y	1	B-Live modules connected to PC, wheelchair computer, etc.
19	Biomotion	pilot	25	n	1	Mote + PC
20	AID-N	pilot	10	y	1+	Mote + PC
21	Firefighting	prototype	20	y	1+	?
22	Rehabil	prototype	?	y	1	Mote + PC
23	CargoNet	pilot	<10	n	1+	Mote + PC?
24	Fence monitor	prototype	10	n	1	Mote + PC?
25	BikeNet	prototype	5	n	7+	802.15.4/Bluetooth bridge + Nokia N80 OR mote + Aruba AP-70 embedded PC
26	BriMon	prototype	12	n	1	Mobile train TMote, static bridge Tmotes
27	IP net	pilot	25	n	1	Mote + PC?
28	Smart home	prototype	12	y	1	Embedded PC with touchscreen, internet, wall powered
29	SVATS	prototype	6	n	1	?
30	Hitchhiker	pilot?	16		1	?
31	Daily morning	prototype	1	n	1	Mote + MIB510 board + PC
32	Heritage	stable	17	y	1	3Mate mote + Gumstix embedded PC with SD card and WiFi
33	AC meter	pilot	49	n	2+	Meraki Mini and the OpenMesh Mini-Router wired together with radio
34	Coal mine	prototype	27	n	1	?
35	ITS	prototype	8	n	1	?
36	Underwater	prototype	4	n	0	-
37	PipeProbe	prototype	1	n	1	Mote + PC
38	Badgers	stable	74	y	1+	Mote
			mobile + 26?			
			static			
39	Helens volcano	pilot	13	n	1	?
40	Tunnels	pilot	40	n	2	Mote + Gumstix Verdex Pro

Deployment state represents maturity of the application: whether it is a prototype or a pilot test-run in a real environment or it has been running in a stable state for a while. As can be seen, only a few deployments are in a stable state; the majority are prototypes and pilot studies. Therefore, it is important to support fast prototyping and effective debugging mechanisms for these phases.

Despite theoretical assumptions about huge networks consisting of thousands of nodes, only a few deployments contain more than 100 nodes. Eighty percent of listed deployments contain 50 or less nodes, 34%: less than 10 nodes (Figure 1). It seems that the most active period of large-scale WSN deployment has been experienced in the years 2004–2006, with networks consisting of 100 and more nodes (Figure 2).

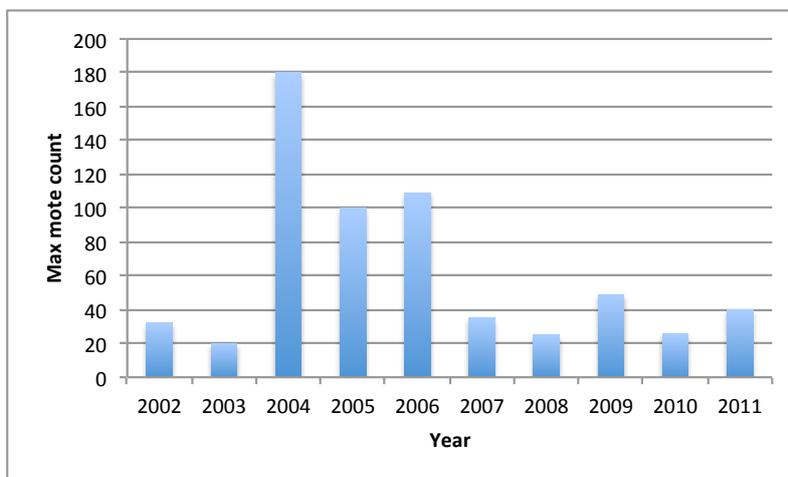
Figure 1. Distribution function of mote count in surveyed deployments—Eighty percent of deployments contain less than 50 motes; 50%: less than 20 motes; and 34%: ten or less.



Design rule 1: The communication stack included in the default OS libraries should concentrate on usability, simplicity and resource efficiency, rather than providing complex and resource-intensive, scalable protocols for thousands of nodes.

Another theoretical assumption, which is only partially true, is a heterogeneous network. The majority of deployments are built on homogenous networks with equal nodes: 70% of deployments. However, significant amount of deployments contain heterogeneous nodes, and that must be taken into account in remote reprogramming design. Remote reprogramming is essential, as it is very time-intensive and difficult to program even more than five nodes. Additionally, often, nodes need many reprogramming iterations after initial setup at the deployment site. Users must be able to select subsets of network nodes to reprogram. Different node hardware must be supported in a single network.

Figure 2. Maximum mote count in surveyed deployments, in each year— peak size in the years 2004–2006; over 100 motes used.



Although remote reprogramming is a low-level function, it can be considered as a debug phase feature, and external tools, such as QDiff [57], can be used to offload this responsibility from the operating system.

Almost all (95%) networks have a sink node or base station, collecting the data. A significant part of deployments use multiple sinks.

Design rule 2: Sink-oriented protocols must be provided and, optionally, multiple sink support.

Almost half of deployments use a regular mote connected to a PC (usually a laptop) as a base station hardware solution.

Design rule 3: The OS toolset must include a default solution for base station application, which is easily extensible to user specific needs.

3.2. Sensing

Table 3 lists the sensing subsystem and sampling characteristics used in deployments.

Table 3. Deployments: sensing.

Nr	Codename	Sensors	Sampling rate, Hz	GPS used
1	Habitats	temperature, light, barometric pressure, humidity and passive infrared	0.0166667	n
2	Minefield	sound, magnetometer, accelerometers, voltage and imaging	?	y
3	Battlefield	magnetometer, acoustic and light	10	n
4	Line in the sand	magnetometer and radar	?	n
5	Counter-sniper	sound	1,000,000	n
6	Electro-shepherd	temperature	?	y
7	Virtual fences	-	?	y

Table 3. *Cont.*

Nr	Codename	Sensors	Sampling rate, Hz	GPS used
8	Oil tanker	accelerometer	19,200	n
9	Enemy vehicles	magnetometer and ultrasound transceiver	?	y, on powered nodes
10	Trove game	accelerometers and light	?	n
11	Elder RFID	RFID reader	1	n
12	Murphy potatoes	temperature and humidity	0.0166667	n
13	Firewxnet	temperature, humidity, wind speed and direction	0.8333333	n
14	AlarmNet	motion, blood pressure, body scale, dust, temperature and light	≤ 1	n
15	Ecuador Volcano	seismometers and acoustic	100	y, on BS
16	Pet game	temperature, light and sound	configurable	n
17	Plug	sound, light, electric current, voltage, vibration, motion and temperature	8,000	n
18	B-Live	light, electric current and switches	?	n
19	Biomotion	accelerometer, gyroscope and capacitive distance sensor	100	n
20	AID-N	pulse oximeter, Electrocardiogram (ECG), blood pressure and heart beat	depends on queries	n
21	Firefighting	temperature	?	n
22	Rehabil	temperature, humidity and light	?	n
23	CargoNet	shock, light, magnetic switch, sound, tilt, temperature and humidity	0.0166667	n
24	Fence monitor	accelerometer	10	n
25	BikeNet	magnetometer, pedal speed, inclinometer, lateral tilt, Galvanic Skin Response (GSR) stress, speedometer, CO ₂ , sound and GPS	configurable	y
26	BriMon	accelerometer	0.6666667	n
27	IP net	temperature, luminosity, vibration, microphone and movement detector	?	n
28	Smart home	Light, temperature, humidity, air pressure, acceleration, gas leak and motion	?	n
29	SVATS	radio Received Signal Strength Indicator (RSSI)	?	n
30	Hitchhiker	air temperature and humidity, surface temperature, solar radiation, wind speed and direction, soil water content and suction and precipitation	?	n
31	Daily morning	accelerometer	50	n
32	Heritage	fiber optic deformation, accelerometers and analog temperature	200	n
33	AC meter	current	$\leq 14,000$	n
34	Coal mine	- (sense radio neighbors only)	-	n
35	ITS	anisotropic magneto-resistive and pyroelectric	varies	n
36	Underwater	pressure, temperature, CDOM, salinity, dissolved oxygen and cameras; motor actuator	≤ 1	n
37	PipeProbe	gyroscope and pressure	33	n
38	Badgers	humidity and temperature	?	n
39	Helens volcano	geophone and accelerometer	100,000?	y

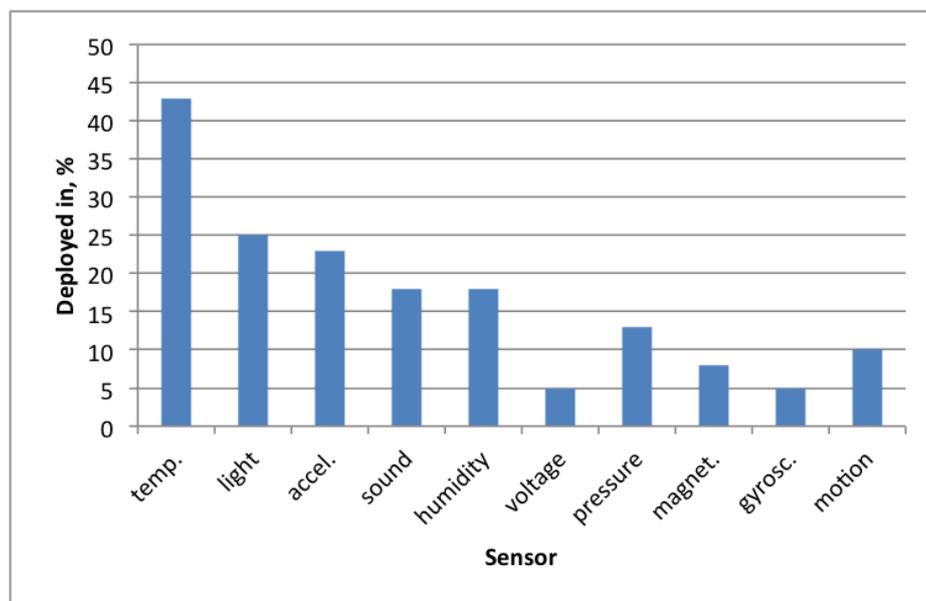
Table 3. *Cont.*

Nr	Codename	Sensors	Sampling rate, Hz	GPS used
40	Tunnels	light, temperature and voltage	0.0333333	n

The most popular sensors are temperature, light and accelerometer sensors (Figure 3).

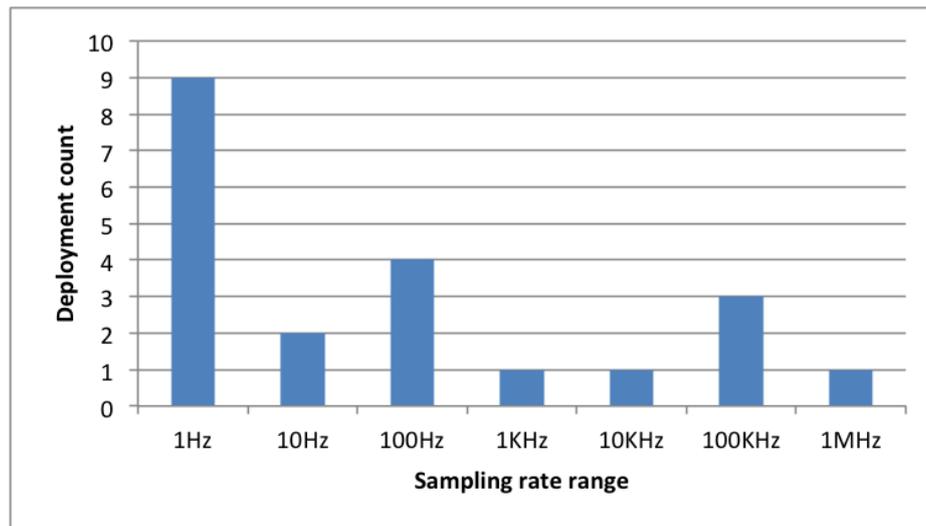
Design rule 4: The WSN operating system should include an Application Programming Interface (API) for temperature, light and acceleration sensors in the default library set.

Figure 3. Sensors used in deployments—Temperature, light and acceleration sensors are the most popular: each of them used in more than 20% of analyzed deployments.



When considering sensor sampling rate, a pattern can be observed (Figure 4). Most of the deployments are low sampling rate examples, where the mote has a very low duty cycle and the sampling rate is less than 1 Hz. Other, less popular application classes use sampling in the range 10–100 Hz and 100–1,000 kHz. The former class uses accelerometer data processing, while the latter is mainly representative of audio and high sensitivity vibration processing. A significant part of applications have a variable sampling rate, configurable in run time.

Figure 4. Sensor sampling rate used in deployments—Low duty cycle applications with sampling rate below 1 Hz are the most popular; however, high-frequency sampling is also used; the ranges 10–100 Hz and 10–100 kHz are popular.



Design rule 5: The operating system must set effective low-frequency, low duty-cycle sampling as the first priority. High performance for sophisticated audio signal processing and other high-frequency sampling applications is secondary, yet required.

GPS localization is a widely used technology, in general; however, it is not very popular in sensor networks, mainly due to unreasonably high power consumption. It is used in less than 18% of deployments. A GPS module should not be considered as a default component.

3.3. Lifetime and Energy

Table 4 describes energy usage and the target lifetime of the analyzed deployments.

Table 4. Deployments: lifetime and energy.

Nr	Codename	Lifetime, days	Energy source	Sleep time, sec	Duty cycle, %	Powered-motes present
1	Habitats	270	battery	60	?	yes, gateways
2	Minefield	?	battery	?	?	yes, all
3	Battlefield	5–50	battery	varies	varies	yes, base station
4	Line in the sand	?	battery and solar	?	?	yes, root
5	Counter-sniper	?	battery	0	100	no
6	Electro-shepherd	50	battery	?	< 1	no
7	Virtual fences	2 h 40 min	battery	0	100	no
8	Oil tanker	82	battery	64,800	< 1	yes, gateways
9	Enemy vehicles	?	battery	?	?	yes, mobile nodes
10	Trove game	?	battery	?	?	yes, base station
11	Elder RFID	?	battery	0?	100?	yes, base station
12	Murphy potatoes	21	battery	60	11	yes, base station

Table 4. Cont.

Nr	Codename	Lifetime, days	Energy source	Sleep time, sec	Duty cycle, %	Powered-motes present
13	Firewxnet	21	battery	840	6.67	yes, gateways
14	AlarmNet	?	battery	?	configurable	yes, base stations
15	Ecuador Volcano	19	battery	0	100	yes, base station
16	Pet game	?	battery	?	?	yes, base station
17	Plug	-	power-net	0	100	yes, all
18	B-Live	-	battery	0	100	yes, all
19	Biomotion	5 h	battery	0	100	yes, base stations
20	AID-N	6	battery	0	100	yes, base station
21	Firefighting	4+	battery	0	100	yes, infrastructure motes
22	Rehabil	?	battery	?	?	yes, base station
23	CargoNet	1825	battery	varies	0.001	no
24	Fence monitor	?	battery	1	?	yes, base station
25	BikeNet	?	battery	?	?	yes, gateways
26	BriMon	625	battery		0.55	no
27	IP net	?	battery	?	20	yes, base station
28	Smart home	?	battery	?	?	yes
29	SVATS	unlimited	power-net	not implemented	-	yes, all
30	Hitchhiker	60	battery and solar	5	10	yes, base station
31	Daily morning	?	battery	0?	100?	yes, base station
32	Heritage	525	battery	0.57	0.05	yes, base station
33	AC meter	?	power-net	?	?	yes, gateways
34	Coal mine	?	battery	?	?	yes, base station?
35	ITS	?	power-net?	0?	100?	yes, all
36	Underwater	?	battery	?	?	no
37	PipeProbe	4 h	battery	0	100	yes, base station
38	Badgers	7	battery	?	0.05	no
39	Helens volcano	400	battery	0?	100?	yes, all
40	Tunnels	480	battery	0.25	?	yes, base stations

Target lifetime is very dynamic among applications, from several hours to several years. Long-living deployments use a duty-cycle below 1%, meaning that sleep mode is used 99% of the time. Both, very short and very long, sleeping periods are used: from 250 milliseconds up to 24 hours.

Operating systems should provide effective routines for duty-cycling and have low computational overhead.

A significant part of deployments (more than 30%), especially in the prototyping phase, do not concentrate on energy efficiency and use a 100% duty cycle.

Design rule 6: The option, “automatically activate sleep mode whenever possible”, would decrease the complexity and increase the lifetime for deployments in the prototyping phase and also help beginner sensor network programmers.

Although energy harvesting is envisioned as the only way for sustainable sensing systems [58], power sources other than batteries or static power networks are rarely used (5% of analyzed deployments). Harvesting module support at the operating system level is, therefore, not an essential part of deployments, until today. However, harvesting popularity may increase in future deployments, and support for it at the OS level could be a valuable research direction.

More than 80% of deployments have powered motes present in the network: at least one node has an increased energy budget. Usually, these motes are capable of running at 100% duty cycle, without sleep mode activation.

Design rule 7: Powered mote availability should be considered when designing a default networking protocol library.

3.4. Sensor Mote

Table 5 lists used motes, radio (or other communication media) chips and protocols.

Table 5. Deployments: used motes and radio chips.

Nr	Codename	Mote	Ready custom	or	Mote motivation	Radio chip	Radio protocol
1	Habitats	Mica	adapted		custom Mica weather board and packaging	RFMonolithics TR1001	?
2	Minefield	WINS NG 2.0 [59]	custom		need for high performance	?	?
3	Battlefield	Mica2	adapted		energy and bandwidth efficient; simple and flexible	Chipcon CC1000	SmartRF
4	Line in the sand	Mica2	adapted		?	Chipcon CC1000	SmartRF
5	Counter-sniper	Mica2	adapted		?	Chipcon CC1000	SmartRF
6	Electro-shepherd	Custom + Active RFID tags	custom		packaging adapted to sheep habits	unnamed Ultra High Frequency (UHF) transceiver	?
7	Virtual fences	Zaurus PDA	ready		off-the-shelf	unnamed WiFi	802.11
8	Oil tanker	Intel Mote	adapted		?	Zeevo TC2001P	Bluetooth 1.1
9	Enemy vehicles	Mica2Dot	adapted		?	Chipcon CC1000	SmartRF
10	Trove game	Mica2	ready		off-the-shelf	Chipcon CC1000	SmartRF
11	Elder RFID	Mica2	adapted		off-the-shelf; RFID reader added	Chipcon CC1000 + RFID	SmartRF + RFID
12	Murphy potatoes	TNode, Mica2-like	custom		packaging + sensing	Chipcon CC1000	SmartRF
13	Firewxnet	Mica2	adapted		Mantis OS [60] support, AA batteries, extensible	Chipcon CC1000	SmartRF
14	AlarmNet	Mica2 + TMote Sky	adapted		off-the-shelf; extensible	Chipcon CC1000	SmartRF
15	Ecuador Volcano	Tmote Sky	adapted		off-the-shelf	Chipcon CC2420	802.15.4
16	Pet game	MicaZ	ready		off-the-shelf	Chipcon CC2420	802.15.4
17	Plug	Plug Mote	custom		specific sensing + packaging	Chipcon CC2500	?
18	B-Live	B-Live module	custom		custom modular system	?	?
19	Biomotion	custom	custom		size constraints	Nordic nRF2401A	-
20	AID-N	TMote Sky + MicaZ	adapted		off-the-shelf; extensible	Chipcon CC2420	802.15.4
21	Firefighting	TMote Sky	adapted		off-the-shelf; easy prototyping	Chipcon CC2420	802.15.4
22	Rehabil	Maxfor TIP 7xxCM: TelosB-compatible	ready		off-the-shelf	Chipcon CC2420	802.15.4
23	CargoNet	CargoNet mote	custom		low power; low cost components	Chipcon CC2500	-
24	Fence monitor	Scatterweb ESB [61]	ready		off-the-shelf	Chipcon CC1020	?

Table 5. Cont.

Nr	Codename	Mote	Ready custom	or	Mote motivation	Radio chip	Radio protocol
25	BikeNet	TMote Invent	adapted		off-the-shelf mote providing required connectivity	Chipcon CC2420	802.15.4
26	BriMon	Tmote Sky	adapted		off the shelf	Chipcon CC2420	802.15.4
27	IP net	Scatterweb ESB	adapted		Necessary sensors on board	TR1001	?
28	Smart home	ZigbeX	custom		specific sensor, size and power constraints	Chipcon CC2420	802.15.4
29	SVATS	Mica2	ready		off-the-shelf	Chipcon CC1000	SmartRF
30	Hitchhiker	TinyNode	adapted		long-range communication	Semtech XE1205	?
31	Daily morning	MicaZ	ready		off-the-shelf	Chipcon CC2420	802.15.4
32	Heritage	3Mate!	adapted		TinyOS supported mote with custom sensors	Chipcon CC2420	802.15.4
33	AC meter	ACme (Epic core)	adapted		modular; convenient prototyping	Chipcon CC2420	802.15.4
34	Coal mine	Mica2	ready		off-the-shelf	Chipcon CC1000	SmartRF
35	ITS	Custom	custom		specific sensing needs	Chipcon CC2420	802.15.4
36	Underwater	AquaNode	custom		specific packaging, sensor and actuator needs	custom	-
37	PipeProbe	Eco mote	adapted		size and energy constraints	Nordic nRF24E1	?
38	Badgers	V1: Tmote Sky + external board; V2: custom	v1: adapted; v2: custom		v1: off-the-shelf v2: optimizations	Atmel AT86RF230	802.15.4
39	Helens volcano	custom	custom		specific computational, sensing and packaging needs	Chipcon CC2420	802.15.4
40	Tunnels	TRITon mote [62] : TelosB-like	custom		reuse and custom packaging	Chipcon CC2420	802.15.4

Mica2 [64] and MicaZ [3] platforms were very popular in early deployments. TelosB-compatible platforms (TMote Sky and others) [2,65] have been the most popular in recent years.

Design rule 8: TelosB platform support is essential.

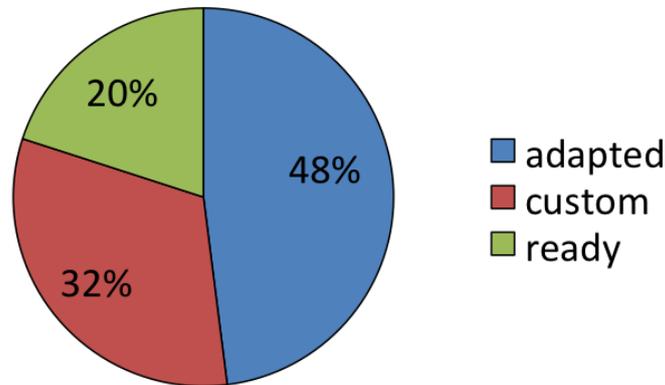
MicaZ support is optional, yet suggested, as sensor network research laboratories might use previously obtained MicaZ motes, especially for student projects.

Almost half of deployments (47%) use adapted versions of off-the-shelf motes by adding customized sensors, actuators and packaging (Figure 5). Almost one third (32%) use custom motes, by combining different microchips. Often, these platforms are either compatible or similar to commercial platforms (for example, TelosB) and use the same microcontrollers (MCUs) and radio chips. Only 20% use motes off-the-shelf with default sensor modules.

Design rule 9: The WSN OS must support implementation of additional sensor drivers for existing commercial motes

Design rule 10: Development of completely new platforms must be simple enough, and highly reusable code should be contained in the OS

Figure 5. Custom, adapted and off-the-shelf mote usage in deployments—Almost half of deployments adapt off-the-shelf motes by custom sensing and packaging hardware, 32% use custom platforms and only 20% use commercial motes with default sensing modules.



The most popular reason for building a customized mote is specific sensing and packaging constraints. The application range is very wide; there will always be applications with specific requirements.

On the other hand, part of the sensor network users are beginners in the field and do not have resources to develop a new platform to assess a certain idea in real-world settings. Off-the-shelf commercial platforms, a simple programming interface, default settings and demo applications are required for this user class.

Chipcon CC1000 [66] radio was popular for early deployments; however, Chipcon CC2420 [67] is the most popular in recent years. IEEE 802.15.4 is the most popular radio transmission protocol (used in CC2420 and other radio chips) at the moment.

Design rule 11: Driver support for CC2420 radio is essential.

More radio chips and system-on-chip solutions using the IEEE 802.15.4 protocol can be expected in the coming years.

3.5. Sensor Mote: Microcontroller

Used microcontrollers are listed in Table 6.

Table 6. Deployments: used microcontrollers (MCUs).

Nr	Codename	MCU count	MCU Name	Architecture, bits	MHz	RAM, KB	Program Memory, KB
1	Habitats	1	Atmel ATmega103L	8	4	4	128
2	Minefield	1	Hitachi SH4 7751	32	167	64,000	0
3	Battlefield	1	Atmel ATmega128	8	7.3	4	128
4	Line in the sand	1	Atmel ATmega128	8	4	4	128
5	Counter-sniper	1 + Field-Programmable Gate Array (FPGA)	Atmel ATmega128L	8	7.3	4	128
6	Electro-shepherd	1	Atmel ATmega128	8	7.3	4	128
7	Virtual fences	1	Intel StrongArm	32	206	65,536	?

Table 6. Cont.

Nr	Codename	MCU count	MCU Name	Architecture, bits	MHz	RAM, KB	Program Memory, KB
8	Oil tanker	1	Zeevo ARM7TDMI	32	12	64	512
9	Enemy vehicles	1	Atmel ATmega128L	8	4	4	128
10	Trove game	1	Atmel ATmega128	8	7.3	4	128
11	Elder RFID	1	Atmel ATmega128	8	7.3	4	128
12	Murphy potatoes	1	Atmel ATmega128L	8	8	4	128
13	Firewxnet	1	Atmel ATmega128L	8	7.3	4	128
14	AlarmNet	1	Atmel ATmega128L	8	7.3	4	128
15	Ecuador Volcano	1	Texas Instruments (TI) MSP430F1611	16	8	10	48
16	Pet game	1	Atmel ATmega128	8	7.3	4	128
17	Plug	1	Atmel AT91SAM7S64	32	48	16	64
18	B-Live	2	Microchip PIC18F2580	8	40	1.5	32
19	Biomotion	1	TI MSP430F149	16	8	2	60
20	AID-N	1	TI MSP430F1611	16	8	10	48
21	Firefighting	1	TI MSP430F1611	16	8	10	48
22	Rehabil	1	TI MSP430F1611	16	8	10	48
23	CargoNet	1	TI MSP430F135	16	8?	0.512	16
24	Fence monitor	1	TI MSP430F1612	16	7.3	5	55
25	BikeNet	1	TI MSP430F1611	16	8	10	48
26	BriMon	1	TI MSP430F1611	16	8	10	48
27	IP net	1	TI MSP430F149	16	8	2	60
28	Smart home	1	Atmel ATmega128	8	8	4	128
29	SVATS	1	Atmel ATmega128L	8	7.3	4	128
30	Hitchhiker	1	TI MSP430F1611	16	8	10	48
31	Daily morning	1	Atmel ATmega128	8	7.3	4	128
32	Heritage	1	TI MSP430F1611	16	8	10	48
33	AC meter	1	TI MSP430F1611	16	8	10	48
34	Coal mine	1	Atmel ATmega128	8	7.3	4	128
35	ITS	2	ARM7 + MSP430F1611	32 + 8	? + 8	64 + 10	? + 48
36	Underwater	1	NXP LPC2148 ARM7TDMI	32	60	40	512
37	PipeProbe	1	Nordic nRF24E1 DW8051	8	16	4.25	32
38	Badgers	1	Atmel ATmega128V	8	8	8	128
39	Helens volcano	1	Intel XScale PXA271	32	13 (624 max)	256	32768
40	Tunnels	1	TI MSP430F1611	16	8	10	48

Only a few deployments use motes with more than one MCU. Therefore, OS support for multi-MCU platforms is an interesting option; however, the potential usage is limited. Multi-MCU motes are a future research area for applications running simple tasks routinely and requiring extra processing power sporadically. Gumsense mote is an example of this approach [68].

The most popular MCUs belong to Atmel ATmega AVR architecture[69] and Texas Instruments MSP430 families. The former is used in Mica-family motes, while the latter is the core of the TelosB platform, which has been widely used recently.

Design rule 12: Support for Atmel AVR and Texas Instruments MSP430 MCU architectures is essential for sensor network operating systems.

Sensor network motes use eight-bit or 16-bit architectures, with a few 32-bit ARM-family exceptions. Typical CPU frequencies are around 8 MHz; RAM amount: 4–10 KB; program memory: 48–128 KB. It must be noted that program memory size is always larger than RAM, sometimes even by a factor of

32. Therefore, RAM memory effective usage is more important, and a reasonable amount of program memory can be sacrificed for that matter.

3.6. Sensor Mote: External Memory

Used external memory characteristics are described in Table 7. While external memory of several megabytes is available on most sensor motes, it is actually seldom used (only in 25% of deployments). Motes often perform either simple decision tasks or forward all the collected data without caching. However, these 25% of deployments are still too many to be completely discarded.

Table 7. Deployments: external memory.

Nr	Codename	Available external memory, KB	Secure (SD)	Digital	External memory used	File system used
1	Habitats	512	n		y	n
2	Minefield	16,000	n		y	y
3	Battlefield	512	n		n	n
4	Line in the sand	512	n		n	?
5	Counter-sniper	512	n		n	n
6	Electro-shepherd	512	n		y	n
7	Virtual fences	?	y		y	y
8	Oil tanker	0	n		n	n
9	Enemy vehicles	512	n		n	n
10	Trove game	512	n		n	n
11	Elder RFID	512	n		n	n
12	Murphy potatoes	512	n		n	n
13	Firewxnet	512	n		n	n
14	AlarmNet	512	n		n	n
15	Ecuador Volcano	1,024	n		y	n
16	Pet game	512	n		n	n
17	Plug	0	n		n	n
18	B-Live	0	n		n	n
19	Biomotion	0	n		n	n
20	AID-N	1,024	n		n	n
21	Firefighting	1,024	n		n	n
22	Rehabil	1,024	n		n	n
23	CargoNet	1,024	n		y	n
24	Fence monitor	0	n		n	n
25	BikeNet	1,024	n		y?	n
26	BriMon	1,024	n		y	n
27	IP net	1,024	n		n	n
28	Smart home	512	n		?	n
29	SVATS	512	n		n	n

Table 7. *Cont.*

Nr	Codename	Available external memory, KB	Secure (SD)	Digital	External memory used	File system used
30	Hitchhiker	1,024	n		n	n
31	Daily morning	512	n		n	n
32	Heritage	1,024	n		n	n
33	AC meter	2,048	n		y	n
34	Coal mine	512	n		n	n
35	ITS	?	n?		?	n?
36	Underwater	2,097,152	y		?	n
37	PipeProbe	0	n		n	n
38	Badgers	2,097,152	y		y	n
39	Helens volcano	0	n		n	n
40	Tunnels	1024	n		n	n

Design rule 13: External memory support for user data storage at the OS level is optional; yet, it should be provided.

Although very popular consumer products, Secure Digital/MultiMediaCard (SD/MMC) cards are even less frequently used (in less than 10% of deployments). The situation is even worse with filesystem use. Despite multiple sensor network filesystems already being proposed previously [70,71], they are seldom used. Furthermore, probably, there is a connection between the (lack of) external memory and filesystem usage—external memories are rarely used, because there is no simple and efficient filesystem for these devices.

Design rule 14: A convenient filesystem interface should be provided by the operating system, so that sensor network users can use it without extra complexity.

3.7. Communication

Table 8 lists deployment communication characteristics.

Table 8. Deployments: communication.

Nr	Codename	Report rate, 1/h	Payload size, B	Radio range, m	Speed, kbps	Connectivity type
1	Habitats	60	?	200 (1,200 with Yagi 12dBi)	40	connected
2	Minefield	?	?	?	?	connected
3	Battlefield	?	?	300	38.4	intermittent
4	Line in the sand	?	1	300	38.4	connected
5	Counter-sniper	?	?	60	38.4	connected
6	Electro-shepherd	0.33	7+	150–200	?	connected
7	Virtual fences	1,800	8?	?	54,000	connected
8	Oil tanker	0.049	?	30	750	connected
9	Enemy vehicles	1,800	?	30	38.4	connected

Table 8. *Cont.*

Nr	Codename	Report rate, 1/h	Payload size, B	Radio range, m	Speed, kbps	Connectivity type
10	Trove game	?	?	?	38.4	connected
11	Elder RFID	?	19	?	38.4	connected
12	Murphy potatoes	6	22		76.8	connected
13	Firewxnet	200	?	400	38.4	intermittent
14	AlarmNet	configurable	29	?	38.4	connected
15	Ecuador Volcano	depends on events	16	1,000	250	connected
16	Pet game	configurable	?	100	250	connected
17	Plug	720	21	?	?	connected
18	B-Live	-	?	?	?	connected
19	Biomotion	360,000	16	15	1,000	connected
20	AID-N	depends on queries	?	66	250	connected
21	Firefighting	?	?	20	250	connected
22	Rehabil	?	12	30	250	connected
23	CargoNet	depends on events	?	?	250	sporadic
24	Fence monitor	?	?	300	76.8	connected
25	BikeNet	opportunistic	?	20	250	sporadic
26	BriMon	62	116	125	250	sporadic
27	IP net	?	?	300	19.2	connected
28	Smart home	?	?	75–100 outdoor/20–30 indoor	250	connected
29	SVATS	?	?	400	38.4	connected
30	Hitchhiker	?	24	500	76.8	connected
31	Daily morning	180,000	2?	100	250	connected
32	Heritage	6	?	125	250	intermittent
33	AC meter	60 default (configurable)	?	125	250	connected
34	Coal mine	?	7	4 m forced, 20 m max	38.4	intermittent
35	ITS	varies	5*n	?	250	connected
36	Underwater	900	11	?	0.3	intermittent
37	PipeProbe	72,000	?	10	1,000	connected
38	Badgers	2,380+	10	1,000	250	connected
39	Helens volcano	configurable	?	9,600	250	connected
40	Tunnels	120	?	?	250	connected

The data report rate varies significantly—some applications report once a day, while others perform real-time reporting at 100 Hz. If we search for connection between Table 3 and Table 8, two conclusions can be drawn: a low report rate is associated with a low duty cycle; yet, a low report rate does not necessarily imply a low sampling rate—high-frequency sampling applications with a low report rate do exist [24,48,49].

Typical data payload size is in the range of 10–30 Bytes. However, larger packets are used in some deployments.

Design rule 15: The default packet size provided by the operating system should be at least 30 bytes, with an option to change this constant easily, when required.

Typical radio transmission ranges are on the order of a few hundred meters. Some deployments use long-range links with more than a 1-km connectivity range.

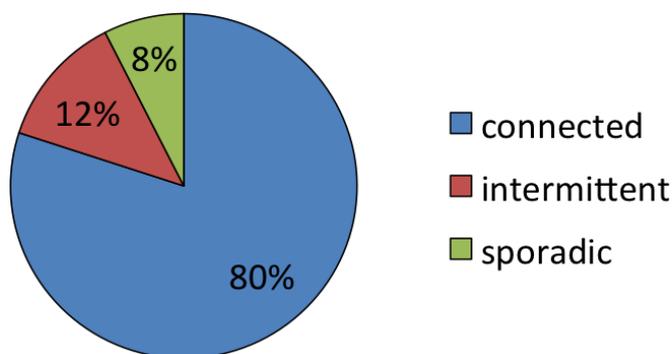
Design rule 16: The option to change radio transmission power (if provided by radio chip) is a valuable option for collision avoidance and energy efficiency.

Design rule 17: Data transmission speed is usually below 1 MBit, theoretically, and even lower, practically. This must be taken into account when designing a communication protocol stack.

Eighty percent of deployments consider the network to be connected without interruptions (Figure 6)—any node can communicate to other nodes at any time (not counting delays imposed by Media Access Control (MAC) protocols). Only 12% experience interruptions, and 8% of networks have only opportunistic connectivity.

Default networking protocols should support connected networks. Opportunistic connection support is optional.

Figure 6. Deployment network connectivity—Eighty percent of deployments consider a network to be continuously connected, while only 12% experience significant disconnections and 8% use opportunistic communication.



3.8. Communication Media

Used communication media characteristics are listed in Table 9.

Table 9. Deployments: communication media.

Nr	Codename	Communication media	Used channels	Directionality used
1	Habitats	radio over air	1	n
2	Minefield	radio over air + sound over air	?	n
3	Battlefield	radio over air	1	n
4	Line in the sand	radio over air	1	n
5	Counter-sniper	radio over air	1	n
6	Electro-shepherd	radio over air	?	n
7	Virtual fences	radio over air	2	y
8	Oil tanker	radio over air	79	n
9	Enemy vehicles	radio over air	1	n
10	Trove game	radio over air	1	n
11	Elder RFID	radio over air	1	n
12	Murphy potatoes	radio over air	1	n
13	Firewxnet	radio over air	1	y, gateways
14	AlarmNet	radio over air	1	n
15	Ecuador Volcano	radio over air	1	y

Table 9. *Cont.*

Nr	Codename	Communication media	Used channels	Directionality used
16	Pet game	radio over air	1	n
17	Plug	radio over air	?	n
18	B-Live	wire mixed with radio over air	?	n
19	Biomotion	radio over air	1	n
20	AID-N	radio over air	1	n
21	Firefighting	radio over air	4	n
22	Rehabil	radio over air	1?	n
23	CargoNet	radio over air	1	n
24	Fence monitor	radio over air	1	n
25	BikeNet	radio over air	1	n
26	BriMon	radio over air	16	n
27	IP net	radio over air	1	n
28	Smart home	radio over air	16	?
29	SVATS	radio over air	?	n
30	Hitchhiker	radio over air	1	n
31	Daily morning	radio over air	1	n
32	Heritage	radio over air	1	n
33	AC meter	radio over air	1	n
34	Coal mine	radio over air	1	n
35	ITS	radio over air	1	n
36	Underwater	ultra-sound over water	1	n
37	PipeProbe	radio over air and water	1	n
38	Badgers	radio over air	?	n
39	Helens volcano	radio over air	1?	y
40	Tunnels	radio over air	2	n

With few exceptions, the communication is performed by transmitting radio signals over air. Ultrasound is used as an alternative. Some networks may use available wired infrastructure.

Eighty-five percent of applications use one, static radio channel; the remaining 15% do switch between multiple alternative channels. If radio channel switching is complex and code-consuming, it should be optional at the OS level.

While directionality usage for extended coverage and energy efficiency has been a widely discussed topic, the ideas are seldom used in practice. Only 10% of deployments use radio directionality benefits, and none of these deployments utilize electronically switchable antennas capable of adjusting directionality in real time [72]. A directionality switching interface is optional; users may implement it in the application layer as needed.

3.9. Network

Deployment networking is summarized in Table 10.

Table 10. Deployments: network.

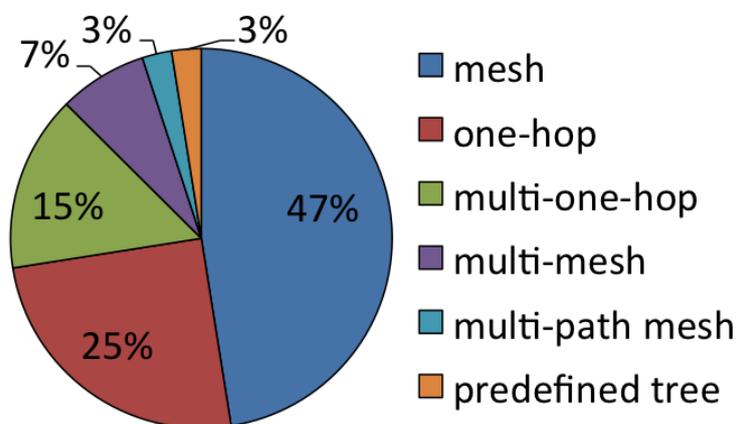
Nr	Codename	Network topology	Mobile notes	Deployment area	Max hop count	Randomly deployed
1	Habitats	multi-one-hop	n	1,000 × 1,000 m	1	n
2	Minefield	multi-one-hop	y	30 × 40 m	?	y
3	Battlefield	multi-one-hop	n	85 m long road	?	y
4	Line in the sand	mesh	n	18 × 8 m	?	n
5	Counter-sniper	multi-one-hop	n	30 × 15 m	11	y
6	Electro-shepherd	one-hop	y	?	1	y (attached to animals)
7	Virtual fences	mesh	y	300 × 300 m	5	y (attached to animals)
8	Oil tanker	multi-one-hop	n	150 × 100 m	1	n
9	Enemy vehicles	mesh	y, power node	20 × 20 m	6	n
10	Trove game	one-hop	y	?	1	y, attached to users
11	Elder RFID	one-hop	n (mobile RFID tags)	< 10 m ²	1	n
12	Murphy potatoes	mesh	n	10,00 × 1,000 m	10	n
13	Firewxnet	multi-mesh	n	160 km ²	4?	n
14	AlarmNet	mesh	y, mobile body notes	apartment	?	n
15	Ecuador Volcano	mesh	n	8,000 × 1,000 m	6	n
16	Pet game	mesh	y	?	?	y
17	Plug	mesh	n	40 × 40	?	n
18	B-Live	multi-one-hop	n	house	2	n
19	Biomotion	one-hop	y, mobile body notes	room	1	n (attached to predefined body parts)
20	AID-N	mesh	y	?	1+	y, attached to users
21	Firefighting	predefined tree	y, human mote	3, 200 m ²	?	n
22	Rehabil	one-hop	y, human notes	gymnastics room	1	y, attached to patients and training machines
23	CargoNet	one-hop	y	truck, ship or plane	1	n
24	Fence monitor	one-hop?	n	35 × 2 m	1?	n
25	BikeNet	mesh	y	5 km long track	?	y (attached to bicycles)
26	BriMon	multi-mesh	y, mobile BS	2,000 × 1	4	n
27	IP net	multi-one-hop	n	250 × 25 3 story building + mock-up town 500 m ²	?	n
28	Smart home	one-hop	n	?	?	n
29	SVATS	mesh	y, notes in cars	parking place	?	n
30	Hitchhiker	mesh	n	500 × 500 m	2?	n
31	Daily morning	one-hop	y, body mote	house	1	n (attached to human)
32	Heritage	mesh	n	7.8 × 4.5 × 26 m	6	n (initial deployment static, but can be moved later)
33	AC meter	mesh	n	building	?	y (Given to users who plug in power outlets of their choice)
34	Coal mine	multi-path mesh	n	8 × 4 × ? m	?	n
35	ITS	mesh	n	140 m long road	7?	n
36	Underwater	mesh	y	?	1	n
37	PipeProbe	one-hop	y	0.18 × 1.40 × 3.45 m	1	n
38	Badgers	mesh	y	1,000 × 2,000 m ?	?	y (attached to animals)
39	Helens volcano	mesh	n	?	1+?	n
40	Tunnels	multi-mesh	n	230 m long tunnel	4	n

A mesh, multi-hop network is the most popular network topology-used in 47% of analyzed cases (Figure 7). The second most popular topology is a simple one-hop network: 25%. Multiple such

one-hop networks are used in 15% of deployments. Altogether, routing is used in 57% of cases. Maximum hop count does not exceed 11 in the surveyed deployments. A rather surprising finding is that almost half of deployments (47%) have at least one mobile node in the network (while maintaining a connected network).

Design rule 18: Multi-hop routing is required as a default component, which can be turned off, if one-hop topology is used. Topology changes must be expected; at least 11 hops should be supported.

Figure 7. Deployment network topologies—Almost half (47%) use a multi-hop mesh network. One-hop networks are used in 25% of cases; 15% use multiple one-hop networks.



Additionally, 30% have random initial node deployment, increasing the need for a neighbor discovery protocol. Neighbor discovery protocols (either explicit or built-in routing) should be provided by the OS.

3.10. In-Network Processing

In-network preprocessing, aggregation and distributed algorithm usage is shown in Table 11 and visualized in Figure 8. Application level aggregation is considered here—data averaging and other compression techniques with the goal to reduce the size of data to be sent.

Table 11. Deployments: in-network processing.

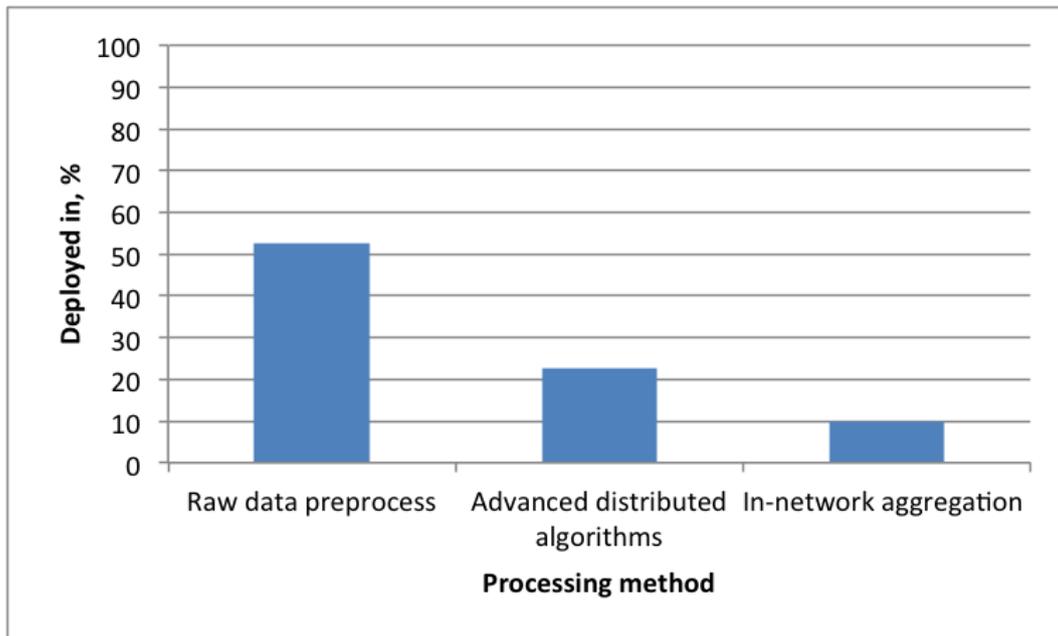
Nr	Codename	Raw data preprocess	Advanced distributed algorithms	In-network aggregation
1	Habitats	n	n	n
2	Minefield	y	y	?
3	Battlefield	y	n	y
4	Line in the sand	y	y	?
5	Counter-sniper	y	n	y
6	Electro-shepherd	n	n	n
7	Virtual fences	n	n	n
8	Oil tanker	n	n	n
9	Enemy vehicles	y	y	y

Table 11. *Cont.*

Nr	Codename	Raw data preprocess	Advanced distributed algorithms	In-network aggregation
10	Trove game	n	y	n
11	Elder RFID	n	n	n
12	Murphy potatoes	n	n	n
13	Firewxnet	n	n	n
14	AlarmNet	y	n	n
15	Ecuador Volcano	y	y	n
16	Pet game	n	n	n
17	Plug	y	n	n
18	B-Live	y	n	n
19	Biomotion	n	n	n
20	AID-N	y	n	n
21	Firefighting	n	n	n
22	Rehabil	n	n	n
23	CargoNet	n	n	n
24	Fence monitor	y	y	n
25	BikeNet	n	n	n
26	BriMon	n	n	n
27	IP net	y	n	n
28	Smart home	y	n	?
29	SVATS	y	y	n
30	Hitchhiker	n	n	n
31	Daily morning	n	n	n
32	Heritage	y	n	n
33	AC meter	y	n	n
34	Coal mine	y	y	y
35	ITS	y	n	n
36	Underwater	n	y	n
37	PipeProbe	y	n	n
38	Badgers	y	n	n
39	Helens volcano	y	n	n
40	Tunnels	n	n	n

As the results show, raw data preprocessing is used in 52% of deployments, *i.e.*, one out of two deployments reports raw data without processing it locally. The situation is even worse with distributed algorithms (voting, distributed motor control, *etc.*) and data aggregation: it is only used in 20% and 10% of cases, respectively. Therefore, sensor network theoretical assumptions, “smart devices taking in-network distributed decisions” and “to save communication bandwidth, aggregation is used”, prove not to be true in reality. Raw data preprocessing and distributed decision-making is performed at the application layer; no responsibility for the operating system is imposed. Aggregation could be performed at the operating system service level. However, it seems that such additional service is not required for most of the applications. Data packet aggregation is optional and should not be included at the OS level.

Figure 8. Deployment in-network processing—Raw data preprocessing is used in half of deployments; distributed algorithms and aggregation are seldom used.



3.11. Networking Stack

The networking protocol stack is summarized in Table 12.

Table 12. Deployments: networking protocol stack.

Nr	Codename	Custom MAC	Channel access method	Routing used	Custom routing	Reactive or proactive routing	IPv6 used	Safe delivery	Data priorities
1	Habitats	n	Carrier Sense Multiple Access (CSMA)	n	-	-	n	n	n
2	Minefield	?	?	?	?	?	?	?	?
3	Battlefield	y	CSMA	y	y	proactive	n	y	?
4	Line in the sand	y	CSMA	y	y	proactive	n	y	n
5	Counter-sniper	n	CSMA	y	y	proactive	n	n	-
6	Electro-shepherd	y	CSMA	-	-	-	n	n	n
7	Virtual fences	n	CSMA	-	-	-	IPv4?	n	n
8	Oil tanker	n	CSMA	-	n	-	n	y	n
9	Enemy vehicles	y	CSMA	y	y	proactive	n	n	-
10	Trove game	n	CSMA	n	-	-	n	n	n
11	Elder RFID	n	CSMA	n	-	-	n	n	n
12	Murphy potatoes	y	CSMA	y	n	proactive	n	n	n
13	Firewxnet	y	CSMA	y	y	proactive	n	y	n
14	AlarmNet	y	CSMA	y	n	?	n	y	y
15	Ecuador Volcano	n	CSMA	y	y	proactive	n	y	n
16	Pet game	n	CSMA	y	n	?	n	n	n
17	Plug	y	CSMA	y	y	?	n	n	n
18	B-Live	?	?	n	-	-	n	?	?

Table 12. *Cont.*

Nr	Codename	Custom MAC	Channel access method	Routing used	Custom routing	Reactive or proactive routing	IPv6 used	Safe delivery	Data priorities
19	Biomotion	y	Time Division Multiple Access (TDMA)	n	-	-	n	n	n
20	AID-N	?	?	y	n	proactive	n	y	n
21	Firefighting	n	CSMA	y, static	n	proactive	n	n	n
22	Rehabil	n	CSMA	n	-	-	n	n	n
23	CargoNet	y	CSMA	n	-	-	n	n	n
24	Fence monitor	n	CSMA?	y	y	proactive?	n	n	n
25	BikeNet	y	CSMA	y	y	reactive	n	y	n
26	BriMon	y	TDMA	y	y	proactive	n	y	n
27	IP net	n	CSMA	y	y	proactive	?	?	?
28	Smart home	?	?	y	?	?	n	?	?
29	SVATS	n	CSMA	y	n	?	n	n	n
30	Hitchhiker	y	TDMA	y	y	reactive	n	y	n
31	Daily morning	n	CSMA	n	-	-	n	n	n
32	Heritage	y	TDMA	y	y	proactive	n	y	y
33	AC meter	n	?	y	n	proactive	y	y	n
34	Coal mine	n	CSMA	y	y	proactive	n	y	n
35	ITS	y?	CSMA?	y	y	reactive	n	y	n
36	Underwater	y	TDMA	n	-	-	n	n	n
37	PipeProbe	n	?	n	-	-	n	n	n
38	Badgers	n	CSMA	y	y	proactive	y	n	y
39	Helens volcano	y	TDMA	y	?	?	n	y	y
40	Tunnels	n	CSMA	y	y	proactive	n	n	n

Forty-three percent of deployments use custom MAC protocols, proving that data link layer problems either really are very application-specific or system developers are not wanting to study the huge amounts of MAC-layer-related published work.

The most commonly used MAC protocols can be divide into two classes: CSMA-based (Carrier Sense Multiple Access) and TDMA-based (Time Division Multiple Access). The former class represents protocols that check media availability shortly before transmission, while in the latter case, all communication participants agree on a common transmission schedule.

Seventy percent use CSMA-based MAC protocols and 15% use TDMA, and the remaining 15% is unclear. CSMA MACs are often used because TDMA implementation is too complex: it requires master node election and time synchronization.

Design rule 19: The operating system should provide a simple, effective and generic CSMA-based MAC protocol by default.

The TDMA MAC option would be a nice feature for the WSN OS, as TDMA protocols are more effective in many cases.

Routing is used in 65% of applications. However, no single best routing protocol is selected—between the analyzed deployment, no two applications used the same routing protocol. Forty-three percent of deployments used custom routing, not published before.

Routing can be proactive: routing tables are prepared and maintained beforehand; or it can be reactive: the routing table is constructed only upon need. The proactive approach is used in 85% of the cases; the remaining 15% use reactive route discovery.

As already mentioned above, the operating system must provide a simple, yet efficient, routing protocol, which performs fair enough for most of the cases. A proactive protocol is preferred.

Design rule 20: The interface for custom MAC and routing protocol substitution must be provided.

Although Internet Protocol version 6 (IPv6) is a widely discussed protocol for the Internet of Things and modifications (such as 6lowpan [73]) for resource-constrained devices have been developed, the protocol is very novel and not widely used yet: only 5% of surveyed deployments use it. However, it can be expected that this number will increase in the coming years. TinyOS [4] and Contiki OS [5] have already included 6lowpan as one of the main networking alternatives.

Design rule 21: It is wise to include a IPv6 (6lowpan) networking stack in the operating system to increase interoperability.

Reliable data delivery is used by 43% of deployments, showing that reliable communication in the transport layer is a significant requirement for some application classes. Another quality-of-service option, data stream prioritizing, is rarely used, though (only 10% of cases).

Design rule 22: Simple transport layer delivery acknowledgment mechanisms should be provided by the operating system.

3.12. Operating System and Middleware

Used operating systems and middleware are listed in Table 13.

Table 13. Deployments: used operating system (OS) and middleware.

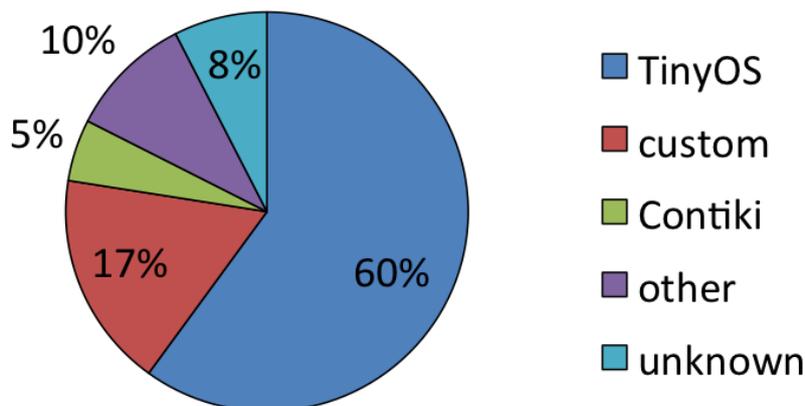
Nr	Codename	OS used	Self-made OS	Middleware used
1	Habitats	TinyOS	n	
2	Minefield	customized Linux	n	
3	Battlefield	TinyOS	n	
4	Line in the sand	TinyOS	n	
5	Counter-sniper	TinyOS	n	
6	Electro-shepherd	?	y	
7	Virtual fences	Linux	n	
8	Oil tanker	?	n	
9	Enemy vehicles	TinyOS	n	
10	Trove game	TinyOS	n	
11	Elder RFID	TinyOS	n	
12	Murphy potatoes	TinyOS	n	

Table 13. *Cont.*

Nr	Codename	OS used	Self-made OS	Middleware used
13	Firewxnet	Mantis OS [60]	y	
14	AlarmNet	TinyOS	n	
15	Ecuador Volcano	TinyOS	n	Deluge [74]
16	Pet game	TinyOS	n	Mate Virtual Machine + TinyScript [75]
17	Plug	custom	y	
18	B-Live	custom	y	
19	Biomotion	no OS	y	
20	AID-N	?	?	
21	Firefighting	TinyOS	n	Deluge [74]?
22	Rehabil	TinyOS	n	
23	CargoNet	custom	y	
24	Fence monitor	ScatterWeb	y	FACTS [76]
25	BikeNet	TinyOS	n	
26	BriMon	TinyOS	n	
27	IP net	Contiki	n	
28	Smart home	TinyOS	n	
29	SVATS	TinyOS?	n	
30	Hitchhiker	TinyOS	n	
31	Daily morning	TinyOS	n	
32	Heritage	TinyOS	n	TeenyLIME [77]
33	AC meter	TinyOS	n	
34	Coal mine	TinyOS	n	
35	ITS	custom?	y?	
36	Underwater	custom	y	
37	PipeProbe	custom	y	
38	Badgers	Contiki	n	
39	Helens volcano	TinyOS	n	customized Deluge [74], remote procedure calls
40	Tunnels	TinyOS	n	TeenyLIME [77]

TinyOS [4] is the de-facto operating system for wireless sensor networks, as is clearly shown in Figure 9: 60% of deployments use it. There are multiple reasons behind that. First, TinyOS has a large community supporting it; therefore, device drivers and protocols are well tested. Second, as it has reached critical mass, TinyOS is the first choice for new sensor network designers—it is being taught at universities, it has easy installation and pretty well developed documentation and even books on how to program in TinyOS [78].

Figure 9. Operating systems used in analyzed deployments—Sixty percent of deployments use the *de-facto* standard: TinyOS. Seventeen percent use self-made or customized OSs.



At the same time, many C and Unix programmers would like to use their previous skills and knowledge to program sensor networks without learning new paradigms, nesC language (used by TinyOS), component wiring, *etc.* One piece of evidence of this statement is that new operating systems for sensor network programming are being developed [5,71,79,80], despite the fact that TinyOS has been here for more than 10 years. Another piece of evidence: in 17% of cases, a self-made or customized OS is used; users either want to use their particular knowledge or they have specific hardware not supported by TinyOS and consider porting TinyOS to new hardware to be too complex.

Deluge [74] and TeenyLIME [77] middleware are used in more than one deployment. Deluge is a remote reprogramming add-on for TinyOS. TeenyLIME is a middleware providing a different level of programming abstraction and, also, implemented on top of TinyOS.

Conclusion: middleware usage is not very popular in sensor networks. Therefore, there is open space for research to develop an easy to use, yet powerful, middleware that is generic enough to be used in a wide application range.

3.13. Software Level Tasks

User and kernel level tasks and services are described in Table 14. The task count and objectives are an estimate of the authors of this deployment survey, developed based on information available from research articles. Networking, time synchronization and remote reprogramming protocols are considered kernel services, if not stated otherwise.

Table 14. Deployments: software level tasks.

Nr	Codename	Kernel service count	Kernel services	App-level task count	App-level tasks
1	Habitats	0		1	sensing + caching to flash + data transfer
2	Minefield	?	linux services	11	
3	Battlefield	2	MAC, routing	2 + 4	Entity tracking, status, middleware (time sync, group management, sentry service, dynamic configuration)
4	Line in the sand	?	?	?	?

Table 14. Cont.

Nr	Codename	Kernel service count	Kernel services	App-level task count	App-level tasks
5	Counter-sniper	?	?	?	?
6	Electro-shepherd	?	-		sense and send
7	Virtual fences	?	MAC	1	sense and issue warning (play sound file)
8	Oil tanker	0		4	cluster formation and time sync, sensing, data transfer
9	Enemy vehicles	?	?	?	?
10	Trove game	1	MAC	3	sense and send, receive, buzz
11	Elder RFID	1	MAC	2	query RFID, report
12	Murphy potatoes	2	MAC, routing	1	sense and send
13	Firewxnet	2	MAC, routing	2	sensing and sending, reception and time-sync
14	AlarmNet	?	?	3	query processing, sensing, report sending
15	Ecuador Volcano	3	time sync, remote reprogram, routing	3	sense, detect events, process queries
16	Pet game	2	MAC, routing	?	sense and send, receive configuration
17	Plug	2	MAC, routing, radio listen	2	sensing and statistics and report, radio RX
18	B-Live	?	?	3	sensing, actuation, data transfer
19	Biomotion	2	MAC, time sync	1	sense and send
20	AID-N	3	MAC, routing, transport	3	query processing, sensing, report sending
21	Firefighting	1	routing	2	sensing and sending, user input processing
22	Rehabil	0?	?	1	sense and send
23	CargoNet	0?	?	1	sense and send
24	Fence monitor	2	MAC, routing	4	sense, preprocess, report, receive neighbor response
25	BikeNet	1	MAC	5	hello broadcast, neighbor discovery and task reception, sensing, data download, data upload
26	BriMon	3	Time sync, MAC, routing	3	sensing, flash storage, sending
27	IP net	?	?	?	?
28	Smart home	?	?	?	?
29	SVATS	2	MAC, time sync	2	listen, decide
30	Hitchhiker	4	MAC, routing, transport, timesync	1	sense and send
31	Daily morning	1	MAC	1	sense and send
32	Heritage			?	?
33	AC meter	?	?	2	sampling, routing
34	Coal mine	2	MAC, routing	2	receive beacons, send beacon and update neighbor map and report accidents
35	ITS	2	MAC, routing	1	listen for queries and sample and process and report
36	Underwater	2	MAC, timesync	3	sensing + sending, reception, motor control
37	PipeProbe	0?	-	1	sense and send
38	Badgers	3	MAC, routing, User Datagram Protocol (UDP) connection establishment	1	sense and send
39	Helens volcano	5	MAC, routing, transport, time sync, remote reprogram	5	sense, detect events, compress, Remote Procedure Call (RPC) response, data report
40	Tunnels	2	MAC, routing	1	sense and send

Most of deployments use not more than two kernel services (55%) (Figure 10). For some deployments, up to five kernel services are used. The maximum service count must be taken into account when designing a task scheduler—if static service maps are used, they must contain enough entries to support all kernel services.

In the application layer, often, just one task is used, which is typically sense and send (33% of cases) (Figure 11). Up to six tasks are used in more complex applications.

Design rule 23: The OS task scheduler should support up to five kernel services and up to six user level tasks. An alternative configuration might be useful, providing a single user task to simplify the

programming approach and provide maximum resource efficiency, which might be important for the most resource-constrained platforms.

Figure 10. The number of kernel level software services used in deployments—fifty-five percent of deployments use two or less kernel services. For 28%, the kernel service count is unknown.

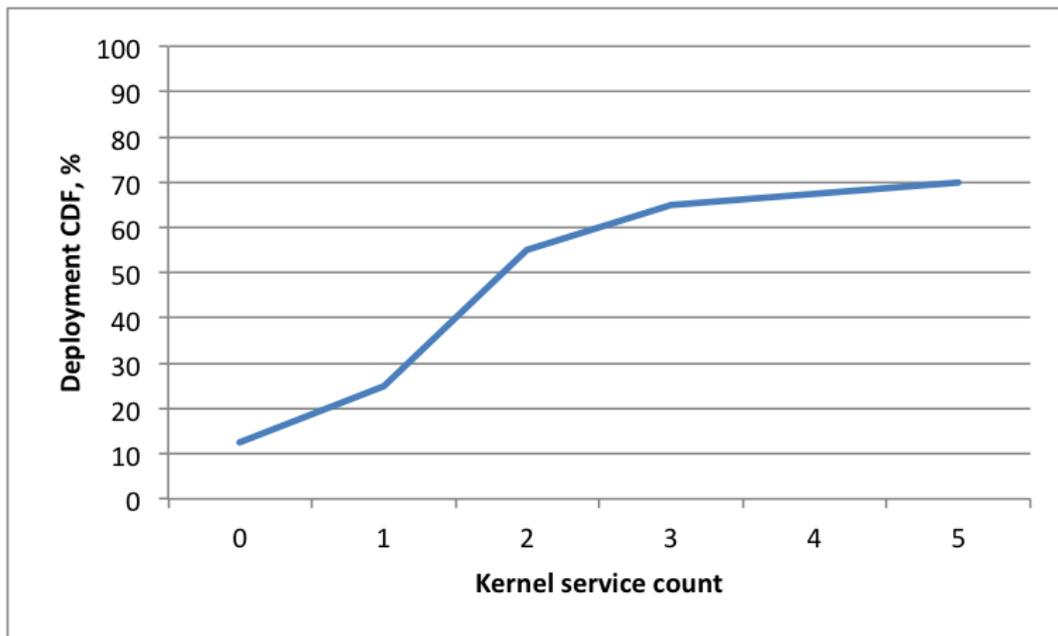
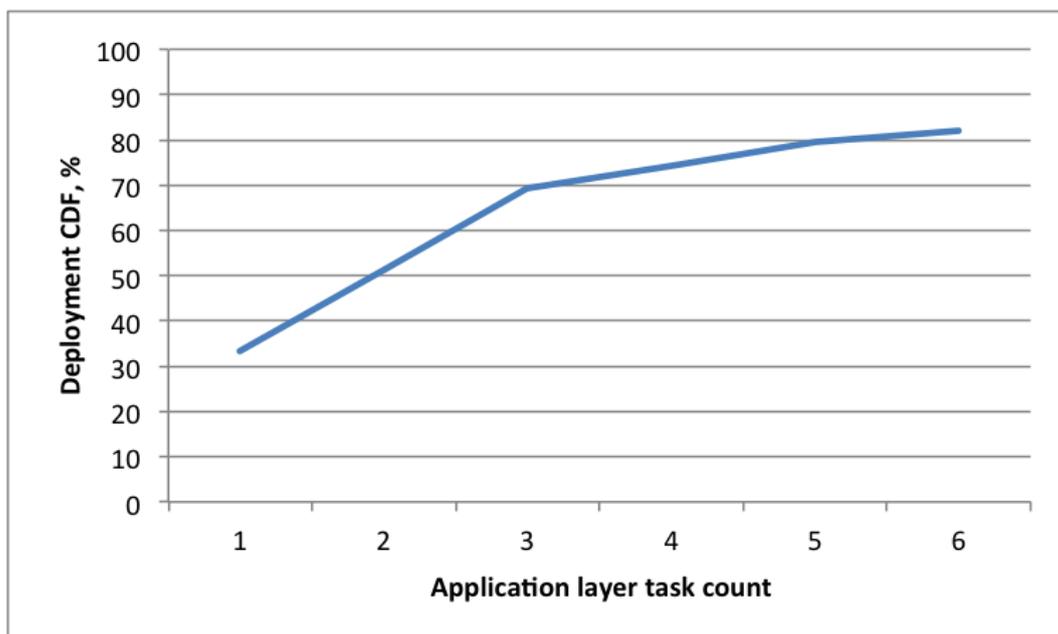


Figure 11. The number of application layer software tasks used in deployments.—Thirty-three percent of deployments use just one task; however, up to six tasks are used in more complex cases. The task count is unknown in 18% of deployments



3.14. Task Scheduling

Table 15 describes deployment task scheduling attributes: time sensitivity and the need for preemptive task scheduling.

Table 15. Deployments: task scheduling.

Nr	Codename	Time sensitive app-level tasks	Preemptive scheduling needed	Task comments
1	Habitats	0	n	sense + cache + send in every period
2	Minefield	7+	y	complicated localization, network awareness and cooperation
3	Battlefield	0	n	
4	Line in the sand	1 ?	n	
5	Counter-sniper	3?	n	localization, synchronization, blast detection
6	Electro-shepherd	?	?	
7	Virtual fences	?	n	
8	Oil tanker	1	y	user-space cluster node discovery and sync are time critical
9	Enemy vehicles	0	n	
10	Trove game	0	n	
11	Elder RFID	0	n	
12	Murphy potatoes	0	n	
13	Firewxnet	1	y	sensing can take up to 200 ms; should be preemptive
14	AlarmNet	0	n	-
15	Ecuador Volcano	1	y	sensing is time-critical, but it is stopped, when the query is received
16	Pet game	0	n	
17	Plug	0	n	
18	B-Live	0	y	
19	Biomotion	0	y	preemption needed for time sync and TDMA MAC
20	AID-N	0	n	
21	Firefighting	0	n	
22	Rehabil	?	?	
23	CargoNet	0	n	wake up on external interrupts; process them; return to sleep mode
24	Fence monitor	0	n	if preprocessing is time-consuming, preemptive scheduling is needed
25	BikeNet	1	y	sensing realized as an app-level TDMA schedule and is time-critical. Data upload may be time-consuming; therefore, preemptive scheduling may be required
26	BriMon	0	n	sending is time critical, but in the MAC layer
27	IP net	0	?	
28	Smart home	?	?	
29	SVATS	0	y	preemption needed for time sync and MAC
30	Hitchhiker	0	y	preemption needed for time sync and MAC
31	Daily morning	0	n	
32	Heritage	1	y	preemptive scheduling needed for time sync?
33	AC meter	0	n	
34	Coal mine	0	n	preemptive scheduling needed, if the neighbor update is time-consuming
35	ITS	0	n	
36	Underwater	0	y	preemption needed for time sync and TDMA MAC
37	PipeProbe	0	n	no MAC; just send
38	Badgers	0	n	
39	Helens volcano	0	y	preemption needed for time sync and MAC
40	Tunnels	0	n	

Two basic scheduling approaches do exist: cooperative and preemptive. In the former case, the switch between tasks is explicit—one task yields a processor to another task. A switch can occur only in predefined code lines. In the latter case, the scheduler can preempt any task at any time and give the CPU to another task. A switch can occur anywhere in the code.

The main advantage of cooperative scheduling is resource efficiency: no CPU time and memory are wasted to perform periodic switches between concurrent tasks, which could be executed serially without any problem.

The main advantage of preemptive scheduling is that users do not have to worry about task switching—it is performed automatically. Even if the user has created an infinite loop in one task, other tasks will have access to the CPU and will be able to execute.

Preemptive scheduling can introduce new bugs, though; it requires context switching, including multiple stack management. Memory checking and overflow control is much harder for multiple stacks, compared to cooperative approaches with a single stack.

If we assume that the user written code is correct, preemptive scheduling is required only in cases where at least one task is time-sensitive and at least one other task is time-intensive (it can execute for a relatively long period of time). The latter may disturb the former from handling all important incoming events.

Twenty percent of analyzed deployments have at least one time-sensitive application layer task (most of them have exactly one), while 30% of deployments require preemptive scheduling. Even in some cases (10%), where no user-space time-sensitive tasks exist, preemption may be required by kernel-level services: MAC protocols and time synchronization.

Design rule 24: The operating system should provide both cooperative and preemptive scheduling, which are switchable as needed.

3.15. Time Synchronization

Time synchronization has been addressed as one of the core challenges of sensor networks. Therefore, its use in deployments is analyzed and statistics are shown in Table 16.

Table 16. Deployments: time synchronization.

Nr	Codename	Time-sync used	Accuracy, μsec	Advanced time-sync	Self-made time-sync
1	Habitats	n	-	-	-
2	Minefield	y	1000	?	?
3	Battlefield	y	?	n	y
4	Line in the sand	y	110	n	y
5	Counter-sniper	y	17.2 (1.6 per hop)	y	y
6	Electro-shepherd	n	-	-	-
7	Virtual fences	n	-	-	-
8	Oil tanker	y	?	n	y
9	Enemy vehicles	n	-	-	-
10	Trove game	n	-	-	-
11	Elder RFID	n	-	-	-
12	Murphy potatoes	n	-	-	-
13	Firewxnet	y	>1000	n	y
14	AlarmNet	n	-	-	-
15	Ecuador Volcano	y	6800	y	n
16	Pet game	n	-	-	-

Table 16. Cont.

Nr	Codename	Time-sync used	Accuracy, μ sec	Advanced time-sync	Self-made time-sync
17	Plug	n	-	-	-
18	B-Live	n	-	-	-
19	Biomotion	y	?	n	y
20	AID-N	n	-	-	-
21	Firefighting	n	-	-	-
22	Rehabil	n	-	-	-
23	CargoNet	n	-	-	-
24	Fence monitor	n	-	-	-
25	BikeNet	y	1 ms?	n, GPS	n
26	BriMon	y	180	n	y
27	IP net	?	?	?	?
28	Smart home	?	?	?	?
29	SVATS	y, not implemented	-	-	-
30	Hitchhiker	y	?	n	y
31	Daily morning	n	-	-	-
32	Heritage	y	732	y	y
33	AC meter	n	-	-	-
34	Coal mine	n	-	-	-
35	ITS	n	-	-	-
36	Underwater	y	?	?	y
37	PipeProbe	n	-	-	-
38	Badgers	n	-	-	-
39	Helens volcano	y	1 ms?	n, GPS	n
40	Tunnels	n	-	-	-

Reliable routing is possible if at least one of two requirements holds:

1. A 100% duty cycle is used on all network nodes functioning as data routers without switching to sleep mode.
2. Network nodes agree on a cooperative schedule for packet forwarding; time synchronization is required.

Therefore, no effective duty cycling and multi-hop routing are possible without time synchronization.

Time synchronization is used in 38% of deployments, while multi-hop routing is used in 57% of cases (the remaining 19% use no duty-cycling).

Although very accurate time synchronization protocols do exist [81], simple methods, including GPS, are used most of the time, offering accuracy in millisecond, not microsecond range.

Only one of deployments used a previously developed time synchronization approach (not including GPS usage in two other deployments); all the others use custom methods. The reason is that despite many published theoretical protocols, no operating system provides an automated and easy way to “switch on” time synchronization.

Design rule 25: Time synchronization provided by the operating system would be of a high value, saving sensor network designers time and effort for custom synchronization development.

3.16. Localization

Another of the most addressed sensor network problems is localization, Table 17.

Table 17. Deployments: localization.

Nr	Codename	Localization used	Localization accuracy, cm	Advanced Localization	Self-made Localization
1	Habitats	n	-	-	-
2	Minefield	y	+/-25	y	y
3	Battlefield	y	couple feet	n	y
4	Line in the sand	n	-	-	-
5	Counter-sniper	y	11	y	y
6	Electro-shepherd	y, GPS	>1 m	n	n
7	Virtual fences	y, GPS	>1 m	n	n
8	Oil tanker	n	-	-	-
9	Enemy vehicles	y	?	n	y
10	Trove game	n	-	-	-
11	Elder RFID	n	-	-	-
12	Murphy potatoes	n	-	-	-
13	Firewxnet	n	-	-	-
14	AlarmNet	y	room	n, motion sensor in rooms	y
15	Ecuador Volcano	n	-	-	-
16	Pet game	n	-	-	-
17	Plug	n	-	-	-
18	B-Live	n	-	-	-
19	Biomotion	n	-	-	-
20	AID-N	n	-	-	-
21	Firefighting	y	<5 m?	n	y
22	Rehabil	n	-	-	-
23	CargoNet	n	-	-	-
24	Fence monitor	n	-	-	-
25	BikeNet	y, GPS	>1 m	n	n
26	BriMon	n	-	-	-
27	IP net	n	-	-	-
28	Smart home	n	-	-	-
29	SVATS	y	?	n, RSSI	y
30	Hitchhiker	n	-	-	-
31	Daily morning	y	room	n	y
32	Heritage	n	-	-	-
33	AC meter	n	-	-	-
34	Coal mine	y	?	n, static	y
35	ITS	y, static	?	n	n
36	Underwater	y	?	n	y
37	PipeProbe	y	8 cm	y	y
38	Badgers	n	-	-	-
39	Helens volcano	n	-	-	-
40	Tunnels	n	-	-	-

Localization is used in 38% of deployments: 8% use GPS and 30%, other methods. In contrast to time synchronization, the localization problem is very application-specific. Required localization granularity, environment, meta-information and infrastructure vary tremendously: in one case, localization of the centimeter scale must be achieved; in another, the room of a moving object must be found; in another, GPS is used in an outdoor environment. In 73% of the cases, where localization is used, it is custom for this application. It is not possible for an operating system to provide a generic localization method for a wide application class. Neighbor discovery service could be usable—it can help to solve both, localization and routing problems.

4. A Typical Wireless Sensor Network

In this section, we present a synthetic example of an average sensor network, based on the most common properties and trends found in the deployment analysis. This example can be used to describe wireless sensor networks to people becoming familiarized with the WSN field.

A typical wireless sensor network:

- is used as a prototyping tool to test new concepts and approaches for monitoring specific environments
- is developed and deployed incrementally in multiple iterations and, therefore, needs effective debugging mechanisms
- contains 10–50 sensor nodes and one or several base stations (a sensor node is connected to a personal computer) that act as data collection sinks
- uses temperature, light and accelerometer sensors
- uses low frequency sensor sampling with less than one sample per second, on average, in most cases; some sensors (accelerometers) require sampling in the range 10–100 Hz, and some scenarios (seismic or audio sensing) use high frequency sampling with a sampling rate above 10 kHz
- has a desired lifetime, varying from several hours (short trials) to several years; relatively often, the desired final lifetime is specified; yet, a significantly shorter lifetime is used in the first proof-of-concept trials with a 100% duty cycle (no sleep mode used)
- has at least one sensor node with increased energy budget—either connected to a static power network or a battery with significantly larger capacity
- has specific sensing and packaging constraints; therefore, packaging and hardware selection are important problems in WSN design
- uses either an adapted version (custom sensors added) of a TelosB-compatible [2] or a MicaZ sensor node [3]; also, fully custom-built motes are popular
- contains MSP430 or AVR architecture microcontrollers on the sensor nodes, typically with eight-bit or 16-bit architecture, 8 MHz CPU frequency, 4–10 KB RAM, 48–128 KB program memory and 512–1,024 KB external memory
- has communication according to the 802.15.4 protocol; TI CC2420 is an example of a widely used wireless communication chip [67]
- sends data packets with a size of 10–30 bytes; the report rate varies significantly—for some scenarios, only one packet per day is sent; for others, each sensor sample is sent at 100 Hz

- uses omnidirectional communication in the range of 100–300 m (each hop) with a transmission speed less than 256 Kbps and uses a single communication channel that can lead to collisions
- considers constant multi-hop connectivity available (with up to 11 hops on the longest route), with possible topology changes, due to mobile nodes or other environmental changes in the sensing region
- has either a previously specified or at least a known sensor node placement (not random)
- is likely to use at least primitive raw data preprocessing before reporting results
- uses CSMA-based MAC protocol and proactive routing, often adapted or completely custom-developed for the particular sensing task
- uses some form of reliable data delivery with acknowledgment reception mechanisms
- has been programmed using the TinyOS operating system
- uses multiple semantically simultaneous application-level tasks, and multiple kernel services are running in background, creating the necessity for effective scheduling mechanisms in the operating system and, also, careful programming of the applications; cooperative scheduling (each task voluntarily yields the CPU to other tasks) is enough in most cases; yet, it requires even more accuracy from the programmers
- requires at least simple time synchronization with millisecond accuracy for common duty cycle management or data time stamping
- may require some form of node localization; yet, the environments pose very specific constraints: indoor/outdoor, required accuracy, update rate, infrastructure availability and many other factors

5. OS Conformance

This section analyzes existing WSN operating system conformance to design rules discussed in this paper. Three operating systems are analyzed here:

- TinyOS [4]—*de facto* standard in the WSN community. Specific environment: event driven programming in nesC language.
- Contiki [5]—more common environment with sequential programming (proto-threads [82]) in American National Standards Institute (ANSI) C programming language
- LiteOS [71]—a WSN OS providing a Unix-like programming interface
- MansOS [84]—a portable, C-based operating system that conforms to most of the design rules described in this paper.

The conformance to the design rules is summarized in Table 18. The following subsections discuss the conformance of the listed operating systems, without describing their structure in detail, as they are already published in other publications [4,5,71,84].

As Table 18 reveals, the listed operating systems cover most of the design rules. Exceptions are discussed here.

Table 18. Existing OS conformance to proposed design rules.

#	Rule	TinyOS	Contiki	LiteOS	MansOS
General					
1	Simple, efficient networking protocols	+	+	±	+
2	Sink-oriented protocols	+	+		+
3	Base station example	+		+	+
Sensing					
4	Temperature, light, acceleration API			±	+
5	Low duty cycle sampling	+	+	+	+
Lifetime and energy					
6	Auto sleep mode	+	+	+	+
7	Powered mode in protocol design	+	+		+
Sensor mote					
8	TelosB support	+	+		+
9	Rapid driver development		+	+	+
10	Rapid platform definition		±		+
11	CC2420 radio chip driver	+	+	+	+
12	AVR and MSP430 architecture support	+	+	±	+
13	External storage support	+	+	+	+
14	Simple file system		+	+	+
Communication					
15	Configurable packet payload (default: 30 bytes)	+	+	+	+
16	Configurable transmission power	+	+	+	+
17	Protocols for ≤ 1 Mbps bandwidth	+	+	+	+
18	Simple proactive routing	+	+	±	+
19	Simple CSMA MAC	+	+		+
20	Custom MAC and routing API	+	+		+
21	IPv6 support	+	+		
22	Simple reception acknowledgment	+	+		+
Tasks and scheduling					
23	five kernel and six user task support	+	+	±	±
24	Cooperative and preemptive scheduling	+	+		+
25	Simple time synchronization		+		+

5.1. TinyOS

TinyOS conforms to the majority of the design rules, but not all of them. The most significant drawback is the complexity of the TinyOS architecture. Although TinyOS is portable (the wide range of supported platforms is a proof for it), code readability and simplicity is doubtful. The main reasons for TinyOS complexity are:

- The event-driven nature: while event handlers impose less overhead compared to sequential programming, with blocking calls and polling, it is more complex for programmers to design and keep in mind the state machine for split-phase operation of the application
- Modular component architecture: a high degree of modularity and code reuse leads to program logic distribution into many components. Each new functionality may require modification in multiple locations, requiring deep knowledge of internal system structure
- nesC language peculiarities: confusion of interfaces and components, component composition and nesting and specific requirements for variable definitions are examples of language aspects interfering with the creativity of novice WSN programmers

These limitations are at the system design level, and there is no quick fix available. The most convenient alternative is to implement middleware on top of TinyOS for simplified access to non-expert WSN programmers. TinyOS architecture is too specific and complex to introduce groundbreaking improvements for readability while maintaining backwards compatibility for existing applications.

There are multiple TinyOS inconsistencies with the proposed design rules, which can be corrected by implementing missing features:

- TinyOS provides an interface for writing data and debug logs to external storage devices; yet, no file system is available. Third party external storage filesystem implementations do exist, such as TinyOS FAT16 support for SD cards [85].
- TinyOS contains Flooding Time Synchronization Protocol (FTSP) time synchronization protocol [9] in its libraries. However, it requires deep understanding of clock skew issues and FTSP protocol operation to be useful
- The temperature, light, acceleration, sound and humidity sensing API is not provided

5.2. Contiki

Contiki is one of the most successful examples regarding conformance to the design rules proposed in this paper.

Contiki does not provide a platform-independent API for popular sensor (temperature, light, sound) and analog-to-digital converter (ADC) access. The reason is that Contiki's mission is not dedicated specifically to sensor networks, but rather to networked embedded device programming. Some of the platforms (such as Apple II) may not have sensors or ADC available; therefore, the API is not explicitly enforced for all the platforms.

Surprisingly, there is no base station application template included. Contiki-collect is provided as an alternative—a complete and configurable sense-and-send network toolset for simple setup of simple sensor network applications.

Portability to new platforms is partially effective. MCU architecture code may be reused. However, the existing approach in Contiki is to copy and duplicate files, even between platforms with a common code base (such as TelosB and Zolertia Z1 [63]). Portability of Contiki can be improved by creating architecture and design guidelines, where a common code base is shared and reused among platforms.

5.3. LiteOS

LiteOS conforms to the proposed design rules only partially.

The LiteOS operating system does not include the networking stack at the OS level. Instead, example routing protocols are implemented at the user level, as application examples. No MAC protocol is available in LiteOS, nor is a unified API for custom MAC and routing protocol development present. The provided routing implements geographic forwarding, without any powered sink node consideration. No IPv6 support or packet reception acknowledgment mechanisms are provided.

Temperature and light sensor reading API is present in LiteOS; the acceleration sensor must be implemented by users.

Only AVR-based hardware platforms are supported, but no TelosB. The source code is, therefore, not optimized for porting to new hardware platforms.

Only preemptive multithreading is available in LiteOS, but no cooperative scheduling. By default, a maximum of eight simultaneous threads are allowed. Additionally, this constant can be changed in the source files. However, each thread requires a separate stack, and running more than eight parallel threads simultaneously on a platform with 4 KiB RAM memory is a rather dangerous experience that can lead to stack overflows and hardly traceable errors. Many parallel task execution is therefore realistic only in scheduling mechanisms sharing stack space between multiple threads.

No time synchronization is included in the LiteOS code base.

5.4. MansOS

MansOS [83] is a portable and easy-to-use WSN operating system that has a smooth learning curve for users with C and Unix programming experience, described in more detail in [84]. One of the main assumptions in MansOS design was the need to adapt it to many different platforms. As the deployment survey shows, this is a very important necessity.

MansOS satisfies all design rules with two exceptions:

- IPv6 support is not built into the MansOS core; it must be implemented at a different level
- MansOS provides both scheduling techniques: preemptive and cooperative. In the preemptive case, only one kernel thread and several user threads are allowed. Multiple kernel tasks must share a single thread in this case. For the cooperative scheduler (*protothreads*, adopted from Contiki [82]), any number of simultaneous threads is allowed, and they all share the same stack space; therefore, the stack overflow probability is significantly lower, compared to LiteOS.

5.5. Summary

The examined WSN operating systems, TinyOS, Contiki, LiteOS and MansOS, conform to the majority of the proposed design rules. However, there is space for improvement for every OS. Some of the drawbacks can be overcome by straight-forward implementation of some missing functionality. However, in some cases, a significant OS redesign is required.

6. Conclusions

This paper surveys 40 wireless sensor network deployments described in the research literature. Based on thorough analysis, design rules for WSN operating system design are proposed. The rules include suggestions related to the task scheduler, networking protocol and other aspects of OS design. Some of the most important concluding design rules:

- In many cases, customized commercial sensor nodes or fully custom-built motes are used. Therefore, OS portability and code reuse are very important.
- Simplicity and extensibility should be preferred over scalability, as existing sensor networks rarely contain more than 100 nodes.
- Both preemptive and cooperative task schedulers should be included in the OS.
- Default networking protocols should be sink-oriented and use CSMA-based MAC and proactive routing protocols. WSN researchers should be able to easily replace default networking protocols with their own to evaluate their performance.
- Simple time synchronization with millisecond (instead of microsecond) accuracy is sufficient for most deployments.

The authors believe that these design rules will foster more efficient, portable and easy-to-use WSN operating system and middleware design.

Another overall conclusion based on analyzed data-existing deployments is rather simple and limited. There is still the need to test larger, more complex and heterogeneous networks in real-world settings. Creation of hybrid networks and “networks of networks” are still open research topics.

Acknowledgments

The authors would like to thank Viesturs Silins for the help in analyzing deployment data and Modris Greitans for providing feedback during the research.

This work has been supported by the European Social Fund, grant Nr. 2009/0138/1DP/1.1.2.1.2/09/IPIA/VIAA/004 “Support for Doctoral Studies at the University of Latvia” and the Latvian National Research Program “Development of innovative multi-functional material, signal processing and information technologies for competitive and research intensive products”.

References

1. Global Security.org. Sound Surveillance System (SOSUS). Available online: <http://www.globalsecurity.org/intell/systems/sosus.htm> (accessed on 8 August 2013).
2. Polastre, J.; Szewczyk, R.; Culler, D. Telos: Enabling Ultra-low Power Wireless Research. In Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, (IPSN'05), UCLA, Los Angeles, CA, USA, 25–27 April 2005.
3. Crossbow Technology. MicaZ mote datasheet. Available online: http://www.openautomation.net/uploads/productos/micaz_datasheet.pdf (accessed on 8 August 2013).

4. Levis, P.; Madden, S.; Polastre, J.; Szewczyk, R.; Whitehouse, K.; Woo, A.; Gay, D.; Hill, J.; Welsh, M.; Brewer, E.; *et al.* Tinyos: An operating system for sensor networks. *Ambient Intell.* **2005**, *35*, 115–148.
5. Dunkels, A.; Gronvall, B.; Voigt, T. Contiki-A Lightweight and Flexible Operating System for Tiny Networked Sensors. In Proceedings of the Annual IEEE Conference on Local Computer Networks, Tampa, FL, USA, 17–18 April 2004; pp. 455–462.
6. Madden, S.; Franklin, M.; Hellerstein, J.; Hong, W. TinyDB: An acquisitional query processing system for sensor networks. *ACM Trans. Database Syst. (TODS)* **2005**, *30*, 122–173.
7. Muller, R.; Alonso, G.; Kossmann, D. A Virtual Machine for Sensor Networks. *ACM SIGOPS Operat. Syst. Rev.* **2007**, *41.3*, 145–158.
8. Demirkol, I.; Ersoy, C.; Alagoz, F. MAC protocols for wireless sensor networks: A survey. *IEEE Commun. Mag.* **2006**, *44*, 115–121.
9. Maróti, M.; Kusy, B.; Simon, G.; Lédeczi, Á. The Flooding Time Synchronization Protocol. In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, (Sensys'04), Baltimore, MD, USA, 3–5 November 2004; pp. 39–49.
10. Mao, G.; Fidan, B.; Anderson, B. Wireless sensor network localization techniques. *Comput. Netw.* **2007**, *51*, 2529–2553.
11. Mainwaring, A.; Culler, D.; Polastre, J.; Szewczyk, R.; Anderson, J. Wireless Sensor Networks for Habitat Monitoring. In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, (WSNA'02), Atlanta, GA, USA, 28 September 2002; pp. 88–97.
12. Merrill, W.; Newberg, F.; Sohrabi, K.; Kaiser, W.; Pottie, G. Collaborative Networking Requirements for Unattended Ground Sensor Systems. In Proceedings of IEEE Aerospace Conference, Big Shq, MI, USA, 8–15 March 2003; pp. 2153–2165.
13. Lynch, J.; Loh, K. A summary review of wireless sensors and sensor networks for structural health monitoring. *Shock Vib. Digest* **2006**, *38*, 91–130.
14. Dunkels, A.; Eriksson, J.; Mottola, L.; Voigt, T.; Oppermann, F.J.; Römer, K.; Casati, F.; Daniel, F.; Picco, G.P.; Soi, S.; *et al.* *Application and Programming Survey*; Technical report, EU FP7 Project makeSense, Swedish Institute of Computer Science: Kista, Sweden, 2010.
15. Mottola, L.; Picco, G.P. Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Comput. Surv.* **2011**, *43*, 19:1–19:51.
16. Bri, D.; Garcia, M.; Lloret, J.; Dini, P. Real Deployments of Wireless Sensor Networks. In Proceedings of SENSORCOMM'09, Athens/Glyfada, Greece, 18–23 June 2009; pp. 415–423.
17. Yick, J.; Mukherjee, B.; Ghosal, D. Wireless sensor network survey. *Comput. Netw.* **2008**, *52*, 2292–2330.
18. Latré, B.; Braem, B.; Moerman, I.; Blondia, C.; Demeester, P. A survey on wireless body area networks. *Wirel. Netw.* **2011**, *17*, 1–18.
19. He, T.; Krishnamurthy, S.; Stankovic, J.A.; Abdelzaher, T.; Luo, L.; Stoleru, R.; Yan, T.; Gu, L.; Hui, J.; Krogh, B. Energy-efficient Surveillance System Using Wireless Sensor Networks. In Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services, (MobiSys'04), Boston, MA, USA, 6–9 June 2004; pp. 270–283.

20. Arora, A.; Dutta, P.; Bapat, S.; Kulathumani, V.; Zhang, H.; Naik, V.; Mittal, V.; Cao, H.; Demirbas, M.; Gouda, M.; *et al.* A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Comput. Netw.* **2004**, *46*, 605–634.
21. Simon, G.; Maróti, M.; Lédeczi, A.; Balogh, G.; Kusy, B.; Nádas, A.; Pap, G.; Sallai, J.; Frampton, K. Sensor Network-based Countersniper System. In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, (SenSys'04), Baltimore, MD, USA, 3–5 November 2004; pp. 1–12.
22. Thorstensen, B.; Syversen, T.; Bjørnvold, T.A.; Walseth, T. Electronic Shepherd—a Low-cost, Low-bandwidth, Wireless Network System. In Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services, (MobiSys'04), Boston, MA, USA, 6–9 June 2004; pp. 245–255.
23. Butler, Z.; Corke, P.; Peterson, R.; Rus, D. Virtual Fences for Controlling Cows. In Proceedings of the 2004 IEEE International Conference on Robotics and Automation, (ICRA'04), Barcelona, Spain, 18–22 April 2004; Volume 5, pp. 4429–4436.
24. Krishnamurthy, L.; Adler, R.; Buonadonna, P.; Chhabra, J.; Flanigan, M.; Kushalnagar, N.; Nachman, L.; Yarvis, M. Design and Deployment of Industrial Sensor Networks: Experiences from a Semiconductor Plant and the North Sea. In Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, (SenSys'05), San Diego, CA, USA, 2–4 November 2005; pp. 64–75.
25. Sharp, C.; Schaffert, S.; Woo, A.; Sastry, N.; Karlof, C.; Sastry, S.; Culler, D. Design and Implementation of a Sensor Network System for Vehicle Tracking and Autonomous Interception. In Proceedings of the Second European Workshop on Wireless Sensor Networks, Istanbul, Turkey, 31 January–2 February 2005; pp. 93–107.
26. Mount, S.; Gaura, E.; Newman, R.M.; Beresford, A.R.; Dolan, S.R.; Allen, M. Trove: A Physical Game Running on an Ad-hoc Wireless Sensor Network. In Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-Aware Services: Usages and Technologies, (sOc-EUSAI'05), Grenoble, France, 12–14 October 2005; pp. 235–239.
27. Ho, L.; Moh, M.; Walker, Z.; Hamada, T.; Su, C.F. A Prototype on RFID and Sensor Networks for Elder Healthcare: Progress Report. In Proceedings of the 2005 ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis, (E-WIND'05), Philadelphia, PA, USA, 22 August 2005; pp. 70–75.
28. Langendoen, K.; Baggio, A.; Visser, O. Murphy Loves Potatoes: Experiences from a Pilot Sensor Network Deployment in Precision Agriculture. In Proceedings of the 20th International IEEE Parallel and Distributed Processing Symposium, (IPDPS 2006), Rhodes Island, Greece, 25–29 April 2006; pp. 1–8.
29. Hartung, C.; Han, R.; Seielstad, C.; Holbrook, S. FireWxNet: A Multi-tiered Portable Wireless System for Monitoring Weather Conditions in Wildland Fire Environments. In Proceedings of the 4th International Conference on Mobile Systems, Applications and Services, (MobiSys'06), Uppsala, Sweden, 19–22 June 2006; pp. 28–41.

30. Wood, A.; Virone, G.; Doan, T.; Cao, Q.; Selavo, L.; Wu, Y.; Fang, L.; He, Z.; Lin, S.; Stankovic, J. *ALARM-NET: Wireless Sensor Networks for Assisted-Living and Residential Monitoring*; Technical Report; University of Virginia Computer Science Department, Charlottesville, VA, USA, 2006.
31. Werner-Allen, G.; Lorincz, K.; Johnson, J.; Lees, J.; Welsh, M. Fidelity and Yield in a Volcano Monitoring Sensor Network. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation, (OSDI'06)*, Seattle, WA, USA, 6–8 November 2006; pp. 381–396.
32. Liu, L.; Ma, H. Wireless Sensor Network Based Mobile Pet Game. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games, (NetGames'06)*, Singapore, Singapore, 30–31 October 2006.
33. Lifton, J.; Feldmeier, M.; Ono, Y.; Lewis, C.; Paradiso, J.A. A Platform for Ubiquitous Sensor Deployment in Occupational and Domestic Environments. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks, (IPSN'07)*, Cambridge, MA, USA, 25–27 April 2007; pp. 119–127.
34. Santos, V.; Bartolomeu, P.; Fonseca, J.; Mota, A. B-Live-a Home Automation System for Disabled and Elderly People. In *Proceedings of the International Symposium on Industrial Embedded Systems, (SIES'07)*, Lisbon, Portugal, 04–06 July 2007; pp. 333–336.
35. Aylward, R.; Paradiso, J.A. A Compact, High-speed, Wearable Sensor Network for Biomotion Capture and Interactive Media. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks, (IPSN'07)*, Cambridge, MA, USA, 25–27 April 2007; pp. 380–389.
36. Gao, T.; Massey, T.; Selavo, L.; Crawford, D.; Chen, B.; Lorincz, K.; Shnayder, V.; Hauenstein, L.; Dabiri, F.; Jeng, J.; *et al.* The advanced health and disaster aid network: A light-weight wireless medical system for triage. *IEEE Trans. Biomed. Circuits Syst.* **2007**, *1*, 203–216.
37. Wilson, J.; Bhargava, V.; Redfern, A.; Wright, P. A Wireless Sensor Network and Incident Command Interface for Urban Firefighting. In *Proceedings of the 4th Annual International Conference on Mobile and Ubiquitous Systems: Networking Services, (MobiQuitous'07)*, Philadelphia, PA, USA, 6–10 August 2007; pp. 1–7.
38. Jarochowski, B.; Shin, S.; Ryu, D.; Kim, H. Ubiquitous Rehabilitation Center: An Implementation of a Wireless Sensor Network Based Rehabilitation Management System. In *Proceedings of the International Conference on Convergence Information Technology, (ICCIT 2007)*, Gyeongju, Korea, 21–23 November 2007; pp. 2349–2358.
39. Malinowski, M.; Moskwa, M.; Feldmeier, M.; Laibowitz, M.; Paradiso, J.A. CargoNet: A Low-cost Micropower Sensor Node Exploiting Quasi-passive Wakeup for Adaptive Asynchronous Monitoring of Exceptional Events. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems, (SenSys'07)*, Sydney, Australia, 6–9 November 2007; pp. 145–159.
40. Wittenburg, G.; Terfloth, K.; Villafuerte, F.L.; Naumowicz, T.; Ritter, H.; Schiller, J. Fence Monitoring: Experimental Evaluation of a Use Case for Wireless Sensor Networks. In *Proceedings of the 4th European Conference on Wireless Sensor Networks, (EWSN'07)*, Delft, The Netherlands, 29–31 January 2007; pp. 163–178.

41. Eisenman, S.B.; Miluzzo, E.; Lane, N.D.; Peterson, R.A.; Ahn, G.S.; Campbell, A.T. BikeNet: A mobile sensing system for cyclist experience mapping. *ACM Trans. Sen. Netw.* **2010**, *6*, 1–39.
42. Chebrolu, K.; Raman, B.; Mishra, N.; Valiveti, P.; Kumar, R. Brimon: A Sensor Network System for Railway Bridge Monitoring. In Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys'08), Breckenridge, CO, USA, 17–20 June 2008; pp. 2–14.
43. Finne, N.; Eriksson, J.; Dunkels, A.; Voigt, T. Experiences from Two Sensor Network Deployments: Self-monitoring and Self-configuration Keys to Success. In Proceedings of the 6th International Conference on Wired/wireless Internet Communications, (WWIC'08), Tampere, Finland, 28–30 May 2008; pp. 189–200.
44. Suh, C.; Ko, Y.B.; Lee, C.H.; Kim, H.J. The Design and Implementation of Smart Sensor-based Home Networks. In Proceedings of the International Symposium on Ubiquitous Computing Systems, (UCS'06), Seoul, Korea, 11–13 November 2006; p. 10.
45. Song, H.; Zhu, S.; Cao, G. SVATS: A Sensor-Network-Based Vehicle Anti-Theft System. In Proceedings of the 27th Conference on Computer Communications, (INFOCOM 2008), Phoenix, AZ, USA, 15–17 April 2008; pp. 2128–2136.
46. Barrenetxea, G.; Ingelrest, F.; Schaefer, G.; Vetterli, M. The Hitchhiker's Guide to Successful Wireless Sensor Network Deployments. In Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, (SenSys'08), Raleigh, North Carolina, 5-7 November 2008; pp. 43–56.
47. Ince, N.F.; Min, C.H.; Tewfik, A.; Vanderpool, D. Detection of early morning daily activities with static home and wearable wireless sensors. *EURASIP J. Adv. Signal Process.* **2008**, doi:10.1155/2008/273130.
48. Ceriotti, M.; Mottola, L.; Picco, G.P.; Murphy, A.L.; Guna, S.; Corra, M.; Pozzi, M.; Zonta, D.; Zanon, P. Monitoring Heritage Buildings with Wireless Sensor Networks: The Torre Aquila Deployment. In Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, (IPSN'09), San Francisco, USA, 13-16 April 2009; pp. 277–288.
49. Jiang, X.; Dawson-Haggerty, S.; Dutta, P.; Culler, D. Design and Implementation of a High-fidelity AC Metering Network. In Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, (IPSN'09), San Francisco, CA, USA, 13–16 April 2009; pp. 253–264.
50. Li, M.; Liu, Y. Underground coal mine monitoring with wireless sensor networks. *ACM Trans. Sens. Netw. (TOSN)* **2009**, *5*, 10:1–10:29.
51. Franceschinis, M.; Gioanola, L.; Messere, M.; Tomasi, R.; Spirito, M.; Civera, P. Wireless Sensor Networks for Intelligent Transportation Systems. In Proceedings of the IEEE 69th Vehicular Technology Conference, VTC Spring 2009, Barcelona, Spain, 26–29 April 2009; pp. 1–5.
52. Detweiler, C.; Doniec, M.; Jiang, M.; Schwager, M.; Chen, R.; Rus, D. Adaptive Decentralized Control of Underwater Sensor Networks for Modeling Underwater Phenomena. In Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, (SenSys'10), Zurich, Switzerland, 3–5 November 2010; pp. 253–266.

53. Lai, T.T.T.; Chen, Y.H.T.; Huang, P.; Chu, H.H. PipeProbe: A Mobile Sensor Droplet for Mapping Hidden Pipeline. In Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, (SenSys'10), Zurich, Switzerland, 3–5 November 2010; pp. 113–126.
54. Dyo, V.; Ellwood, S.A.; Macdonald, D.W.; Markham, A.; Mascolo, C.; Pásztor, B.; Scellato, S.; Trigoni, N.; Wotextcolorreders, R.; Yousef, K. Evolution and Sustainability of a Wildlife Monitoring Sensor Network. In Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, (SenSys'10), Zurich, Switzerland, 3–5 November 2010; pp. 127–140.
55. Huang, R.; Song, W.Z.; Xu, M.; Peterson, N.; Shirazi, B.; LaHusen, R. Real-world sensor network for long-term volcano monitoring: Design and findings. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 321–329.
56. Ceriotti, M.; Corrà, M.; D'Orazio, L.; Doriguzzi, R.; Facchin, D.; Guna, S.; Jesi, G.; Cigno, R.; Mottola, L.; Murphy, A.; *et al.* Is There Light at the Ends of the Tunnel? Wireless Sensor Networks for Adaptive Lighting in Road Tunnels. In Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN/SPOTS), Chicago, IL, USA, 12–14 April 2011; pp. 187–198.
57. Shafi, N.B. Efficient Over-the-Air Remote Reprogramming of Wireless Sensor Networks. MS.c Thesis, Queen's University, Kingston, ON, Canada, 2011.
58. Dutta, P. Sustainable sensing for a smarter planet. *XRDS* **2011**, *17*, 14–20.
59. Sensoria. Wireless Integrated Network Sensors (WINS) Next Generation. Technical Report, Defense Advanced Research Projects Agency (DARPA). Available online: <http://www.trabucayre.com/page-tinyos.html> (accessed on 8 August 2013), 2004.
60. Bhatti, S.; Carlson, J.; Dai, H.; Deng, J.; Rose, J.; Sheth, A.; Shucker, B.; Gruenwald, C.; Torgerson, A.; Han, R. MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms. *Mobile Netw. Appl.* **2005**, *10*, 563–579.
61. TU Harburg Institute of Telematics. Embedded Sensor Board. Available online: <http://wiki.ti5.tu-harburg.de/wsn/scatterweb/esb> (accessed on 8 August 2013).
62. Picco, G.P. TRITon: Trentino Research and Innovation for Tunnel Monitoring. Available online: <http://triton.disi.unitn.it/> (accessed on 8 August 2013).
63. Zolertia. Z1 Platform. Available online: <http://www.zolertia.com/ti> (accessed on 8 August 2013).
64. Crossbow Technology. *MICA2 Wireless Measurement System datasheet*. Available online: http://bullseye.xbow.com:81/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf (accessed on 8 August 2013)
65. Lo, B.; Thiemjarus, S.; King, R.; Yang, G. Body Sensor network—A Wireless Sensor Platform for Pervasive Healthcare Monitoring. In Proceedings of the 3rd International Conference on Pervasive Computing, Munich, Germany, 08–13 May 2005; Volume 191, pp. 77–80.
66. Texas Instruments. *CC1000: Single Chip Very Low Power RF Transceiver*. Available online: <http://www.ti.com/lit/gpn/cc1000> (accessed on 8 August 2013).
67. Texas Instruments. *CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*. Available online: <http://www.ti.com/lit/gpn/cc2420> (accessed on 8 August 2013).

68. Martinez, K.; Basford, P.; Ellul, J.; Spanton R. Gumsense-a High Power Low Power Sensor Node. In Proceedings of the 6th European Conference on Wireless Sensor Networks, (EWSN'09), Cork, Ireland, 11–13 February 2009.
69. Atmel Corporation. AVR 8-bit and 32-bit Microcontroller. Available online: <http://www.atmel.com/products/microcontrollers/avr/default.aspx> (accessed on 8 August 2013).
70. Hill, J.; Szewczyk, R.; Woo, A.; Hollar, S.; Culler, D.; Pister, K. System architecture directions for networked sensors. *ACM Sigplan Not.* **2000**, *35*, 93–104.
71. Cao, Q.; Abdelzaher, T.; Stankovic, J.; He, T. The LiteOS Operating System: Towards Unix-Like Abstractions for Wireless Sensor Networks. In Proceedings of the 7th International Conference on Information Processing in Sensor Networks, (IPSN'08), St. Louis, MO, USA, 22–24 April 2008; pp. 233–244.
72. Prieditis, K.; Drikis, I.; Selavo, L. SAnTArray: Passive Element Array Antenna for Wireless Sensor Networks. In Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, (SenSys'10), Zurich, Switzerland, 3–5 November 2010; pp. 433–434.
73. Shelby, Z.; Bormann, C. *6LoWPAN: The Wireless Embedded Internet*; Wiley Publishing: Chippingham, Wiltshire, UK, 2010.
74. Hui, J.W.; Culler, D. The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, (SenSys'10), Zurich, Switzerland, 3–5 November 2010; pp. 81–94.
75. Levis, P.; Culler, D. Mate: A tiny virtual machine for sensor networks. *Sigplan Not.* **2002**, *37*, 85–95.
76. Terfloth, K.; Wittenburg, G.; Schiller, J. FACTS: A Rule-Based Middleware Architecture for Wireless Sensor Networks. In Proceedings of the 1st International Conference on Communication System Software and Middleware (COMSWARE), New Delhi, India, 8–12 January 2006.
77. Costa, P.; Mottola, L.; Murphy, A.L.; Picco, G.P. TeenyLIME: Transiently Shared Tuple Space Middleware for Wireless Sensor Networks. In Proceedings of the International Workshop on Middleware for Sensor Networks, (MidSens'06), Melbourne, Australia, 28 November 2006; pp. 43–48.
78. Levis, P.; Gay, D. *TinyOS Programming*, 1st ed.; Cambridge University Press: New York, NY, USA, 2009.
79. Saruwatari, S.; Suzuki, M.; Morikawa, H. A Compact Hard Real-time Operating System for Wireless Sensor Nodes. In Proceedings of the 2009 Sixth International Conference on Networked Sensing Systems, (INSS'09), Pittsburgh, PA, USA, 17–19 June 2009; pp. 1–8.
80. Eswaran, A.; Rowe, A.; Rajkumar, R. Nano-RK: An Energy-aware Resource-centric RTOS for Sensor Networks. In Proceedings of the 26th IEEE International Real-Time Systems Symposium, (RTSS 2005), Miami, FL, USA, 6–8 December 2005; pp. 265–274.
81. Ganeriwal, S.; Kumar, R.; Srivastava, M.B. Timing-sync Protocol for Sensor Networks. In Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, (SenSys'03), Los Angeles, CA, USA, 5–7 November 2003; pp. 138–149.

82. Dunkels, A.; Schmidt, O.; Voigt, T.; Ali, M. Protothreads: Simplifying Event-Driven Programming of Memory-Constrained Embedded Systems. In Proceedings of SenSys'06, Boulder, CO, USA, 31 October–3 November 2006; pp. 29–42.
83. MansOS—Portable and easy-to-use WSN operating system. Available online: <http://mansos.net> (accessed on 8 August 2013).
84. Elsts, A.; Strazdins, G.; Vihrov, A.; Selavo, L. Design and Implementation of MansOS: A Wireless Sensor Network Operating System. In *Scientific Papers*; University of Latvia: Riga, Latvia, 2012; Volume 787, pp. 79–105.
85. Goavec-Merou, G. SDCard and FAT16 File System Implementation for TinyOS. Available online: <http://www.trabucayre.com/page-tinyos.html> (accessed on 8 August 2013).

© 2013 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/3.0/>).