Chapter 8
WSN Platforms

8.1 Introduction

Previous chapters describe sensor network architecture, communication protocols and various characteristics related to security of WSNs. This chapter provides a short overview of hardware and simulation-based WSN platforms. A basic knowledge about the WSN platforms is necessary to understand physical and computational capabilities and limitations of the technology, to foresee future directions in technology development, and to further develop WSN algorithms and protocols.

While very few research and development results can directly be implemented on real wireless sensor nodes and networks, more often such implementations are first carried out on sensor network simulators. WSN simulators provide test platforms that can save time and cost, expand complexity of deployed networks, and thus verify results before any real hardware nodes are deployed. WSN simulators can verify results in simulation that can be costly, dangerous, or impossible to deploy in real-life, large-scale networks. Also, some applications in harsh environments would be extremely difficult for live testing, and effect on possible node failures is easier to simulate on WSN simulators. Therefore, WSN simulators are an integral part of every research and development effort in this area.

8.2 WSN Hardware Platforms

WSN platforms implement the physical layer of the protocol stack and have the primary goal of gathering multi-modal information about the physical phenomena. The three main components of a WSN hardware platform—namely, sensors which perform the sensing functions, the radio that provides communication and

R.R. Selmic et al., *Wireless Sensor Networks*,
DOI 10.1007/978-3-319-46769-6_8

networking, and the microcontroller which performs the processing—are designed with an aim of minimizing the amount of consumed power. This design objective emanates from the fact that WSNs are usually deployed in environments where human intervention after deployment is not expected, a scenario that rules out the possibility of human-supported battery recharging.

One of the first wireless, low-power, real-time monitoring applications was developed by UC Berkeley and was called Mica motes [12]. Those first sensor nodes were based on low-power, 8-bit or 16-bit microcontrollers with few sensors either attached directly to the networking platform or with modular hardware approach where various sensing modules can be easily attached. Included are Mica and Mica2 wireless sensor networks, followed by IRIS platform, and Cricket system discussed later in the chapter. Since then, researchers are actively exploring WSNs applications in numerous domains and engineers have created more advanced platforms with improved computational capabilities [3] such as WSN nodes running on 32-bit ARM Cortex-M3 microcontrollers [3, 9] that reduces latency and energy consumption for computationally intensive tasks, but have increased energy consumption for traditional sensing applications. Further hardware platform development includes FPGA-based nodes [4, 10, 15, 16], that are modular and can be reconfigured. In particular, [15] presents a WSN node that combines FPGA and System-on-Chip technology and is a low-power, a high-performance, adaptable sensor node. Only recent FPGA could be used in such applications as their power consumption was reduced (initial FPGA were extremely power intensive devices).

In this chapter, we describe the most commonly used WSN hardware platforms that are important from historic/technology development perspective or have a high-market penetration. Table 8.1 shows the described WSN hardware nodes.

8.2.1 IRIS

The IRIS sensor nodes platform, from MEMSIC, Fig. 8.1, is an evolution of the first Mica and Mica2 sensor nodes platform [6, 26]. The nodes, also called Motes, run on a 2.4 GHz low-power Atmel's radio (AT86RF230) that is based on IEEE 802.15.4 standard [26] with transmission data rate of 250 kbps.

IRIS motes have an outdoor line-of-sight range of about 500 m and indoor range of about 50 m. They run on 2 AA batteries with current consumption ranging from 10 to 17 mA depending of transmission power of the radio. Motes are based on Atmel ATmega1281 8-bit microcontroller with 128 kB of Flash memory, 4 kB of EEPROM, and 8 kB is an internal SRAM. This is a low-power, static-operation, advanced RISC architecture microcontroller with selectable various frequency of operation from below 4 MHz and up to 16 MHz.

Table 8.1 Common WSN platforms with their microcontroller units, radio, and sensor components

WSN Node	MCU	Radio	Sensors
IRIS	Atmel ATmega1281, RISC architecture (Tiny OS)	2.4 GHz Atmel AT86RF230	Various sensor boards, communication supported using analog inputs, digital I/O, I2C, SPI, UART.
Digi XBee® ZigBee	No MCU, can communicate with variety of microcontrollers through serial interface	865 MHz to 2.4 GHz	No sensors on the module.
WiSense	TI MSP430 (Linux)	CC1000	Temperature, light, and other custom modules
Intel® Mote 2	Intel PXA271 (Tiny OS or Linux)	CC2420 at 2.4 GHz (IEEE 802.15.4)	Stackable modules, can support image and video processing
Mulle	ARM Cortex-M4 (Contiki OS)	IEEE 802.15.4 at 868 MHz, 900 MHz	Different I/O with 60-pin connector, various sensors can be added
CM30x	Jennic JN5148 RISC wireless microcontroller	IEEE 802.15.4 integrated with MCU	Multiple I/O with 34-pin connector for sensor modules
Fleck3	Atmel ATmega128 (Tiny OS)	Nordic nRF905, ISM band 433/868/915 MHz	Built in temperature and light sensors, can add an expansion board with extra sensors
Cricket	Atmel ATmega128L (Tiny OS)	CC1000	51-pin connector for sensor module
Shimmer wireless node	TI MSP430	TI CC2420 at 2.4 GHz (IEEE 802.15.4)	Sensor modules: 9DoF kinematic, GPS, ECG, etc.
ADVANTICSYS XM1000	TI MSP430	TI CC2420 at 2.4 GHz (IEEE 802.15.4)	Integrated temperature, humidity, and two light sensors

Fig. 8.1 IRIS sensor node from MEMSIC (reproduced by permission of MEMSIC, www.memsic.com)

Motes have 51-pin expansion connector that enables stacking of various sensor boards for different applications such as security and building monitoring, acoustic and vibration monitoring, chemical sensing, and more. Communications with multiple sensors is possible through analog inputs, digital I/O, I2C, SPI, and UART interfaces.

IRIS runs on TinyOS [28], an event-based operating system written in nesC programming language, which is a variation of a C language for embedded systems with small memory storage devices. An event handler processes events as they occur. The operating system consists of components connected with interfaces for various wireless sensor network abstractions such as communication, routing, sensing, and more. Components include the specification with names of their interfaces and the interfaces' implementation. Interfaces have a bidirectional feature that has been specified into a set of commands and a set of events. Components have two forms: modules and configurations. Modules can provide the application code that contains different interfaces. The modules specify what interfaces have been used or provided, and then implement them with the corresponding code. Configurations are used for assembling or linking other components together.

A typical application with IRIS WSN nodes is given in [24] where a passive radar is proposed that monitors for presence of people in indoor environments. The WSN nodes measure the Received Signal Strength Indicator (RSSI) that determines how people's motion affects RSSI at the pre-deployed WSN. The method is based on IRIS node capability to directly measure the received signal strength through its Atmel radio chip. In this case the radio serves dual purpose—provides wireless communication platform and measures received signal strength. In [19], IRIS nodes were used for ambient monitoring application as well as to verify energy modeling in WSNs.

8.2.2 WiSense

WiSense is a low-power, modular WSN platform that is based on the TI MSP430 microprocessor and CC1000 radio technologies operating in sub-GHz frequency range (see Fig. 8.2). The microcontroller, radio, and sensor components are all separated as different hardware modules (boards), with the microcontroller connecting to the radio through the SPI interface and to the sensors through I2C, SPI, and UART interfaces. The microprocessor board is equipped with temperature and light sensors as well as EEPROM. The platform is suitable for fast prototyping in home automation, smart building, and industrial automation applications.

The sensor node runs open-source Linux that implements IEEE 802.15.4 standard. The software and sensor nodes support both Full Function Device (can operate as a router node and can also transmit data from other nodes, thus supporting various network topologies) and Reduced Function Device (only sends information to network, cannot relay data from other nodes) modes.

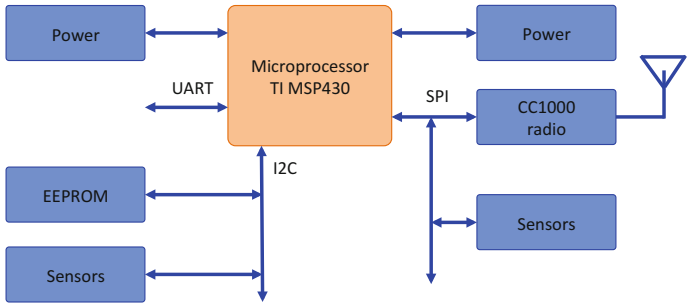

Fig. 8.2 WiSense wireless sensor network node block diagram with main design components

8.2.3 Digi XBee® ZigBee

Digi XBee® ZigBee 802.15.4 RF communication modules are a common platform for wireless connections between various electronic devices, not just sensor nodes, (see Fig. 8.3, [27]). They are a popular platform for wireless sensor networks where sensors can be added as separate components. They operate on frequencies from 865 MHz to 2.4 GHz and support various wireless interfaces including mesh, point-to-point, star, ZigBee, WiFi and others.

Digi XBee® ZigBee operates of 250 kbps data rate and has a range of up to 100 m for 1 mW transmit power devices or 1.6 km for 60 mW transmit power devices. The system supports direct sequence spread spectrum modulation and the 128-bit Advanced Encryption Standard for data encryption. However, MAC-layer addresses are non-encrypted and can be visible to all. The wireless module can operate in a command mode (node firmware can be modified through a set of commands, characters), an idle mode (listens for valid data or RF and serial ports), a receive mode (when a destination node receives a dedicated packet), a transmit mode (prepares to send packets as serial data), and a sleep mode (low-power state when node is not in use, cannot send or receive packets until awaken). Basic commands and changes in the Digi XBee® ZigBee node configuration and firmware can be entered remotely.

Fig. 8.3 Digi XBee® ZigBee wireless communication platform (reproduced by permission of Digi XBee®, http://www.digi.com)

In a typical application for condition monitoring and energy management in homes [7], Digi XBee® ZigBee nodes are used as Internet of Things (IoT) where three different sensors are connected with the coordinator/controller module using ZigBee communication standard in a mesh topology or a star topology (depending of the range between the coordinator and the end nodes). The coordinator is communicating with the gateway that translates the ZigBee protocol to the Internet protocol (IPv6) format. The Digi XBee® ZigBee sensor nodes produce ZigBee packets (64 bits address for a node on a Personal Area Network—PAN and a 16 bit address for the PAN) that are converted into IPv6 packets (128 bits to address a node on the network: 48 bits to address the network, 16 bits to address the PAN, and 64 bits to address the sensor node) and then sent to a central server. In the opposite direction, command packets towards the Digi XBee® ZigBee modules are encapsulated in a User Datagram Protocol (UDP) and then converted back to ZigBee packets by the IoT application gateway. This translation allowed for IoT IPv6 implementation of sensor nodes in 802.15.4 data format and ZigBee network where each sensor node is addressable with its specific IP address [7].

8.2.4 Intel® Mote 2

Intel® Mote 2 or iMote2 [29], from Intel®, is a high-performance WSN node that uses Intel Xscale® PXA271 CPU. It is able to operate at low power since the PXA271 CPU is capable to run at low voltage (0.85 V) and low frequency (13 MHz). The processor has several low-power modes including the sleep and deep sleep modes. In the deep sleep mode it draws a current of about 390 μA (compared to up to 66 mA in active mode). The processor integrates many I/O options that include I2C, 3 Synchronous Serial Ports, 3 high-speed UARTs, fast infrared, camera interface USB client and host and I2S codec audio interfaces among several others. These many I/O options make it very flexible in supporting different sensors. The processor also adds several timers and a real-time clock.

iMote2 integrates the IEEE 802.15.4 radio transceiver (CC2420). This transceiver supports a data rate of 250 kb/s with 16 channels in the 2.4 GHz band. Also, this mote platform integrates a 2.4 GHz surface mount antenna, which has a nominal range of up to 30 m. For applications requiring a longer range, an SMA connector can be soldered directly to the board for the connection of an external antenna. The module includes a power management IC that supplies the processor with the various voltage domains.

For interfacing the sensor board, the iMote2 has two sets of connectors, the basic set and the advanced set. The basic set supports the most common sensor interfaces and can be supported in future mote designs. The advanced set is platform-specific and exposes advanced features such as the camera interface, high-speed bus and audio interfaces. The mote can be powered by primary battery, rechargeable battery, or using mini-B USB connector.

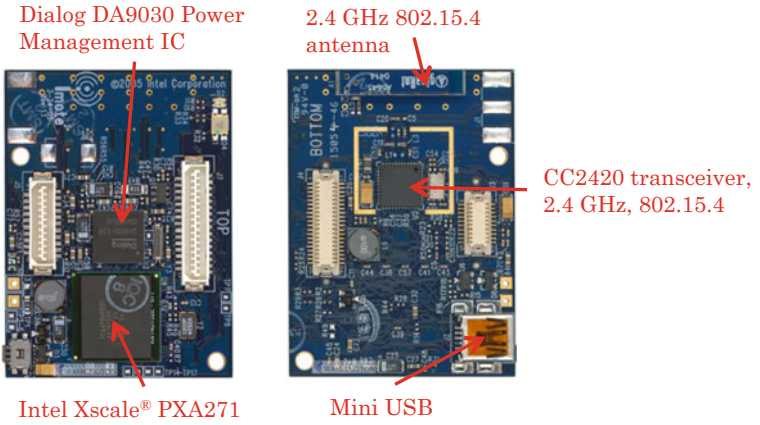

Fig. 8.4 Intel® Mote 2 sensor node, *top view* (*left*) and *bottom view* (*right*) (reproduced by permission from [29])

The iMote2 supports a range of operating systems that include TinyOS for extremely low-power sensor network applications and Linux for more advanced applications. Detailed specifications of this node can be found in [29]. The following figures show the top and bottom of the Intel Mote 2 sensor node (Fig. 8.4).

8.2.5 Mulle

Mulle [30] (from Eistec AB, Fig. 8.5) is a wireless Embedded Internet System (EIS) for wireless sensors connected to the Internet of Things (IoT). Mulle platform comprises the Mulle sensor nodes, the Mulle Internet gateway device and the cloud services and Mulle development tools. The platform uses an ARM Cortex-M4 microcontroller and an IEEE 802.15.4 radio operating at 868 MHz. A frequency of 900 MHz is supported upon request. The Mulle is able to store large amounts of sensor and configuration data on its on-board 2 MB flash memory. It has a high-density expansion port with a 60-pin connector that supports a large number of I/O options (both digital and analog), enabling connectivity to other types of sensors and various debugging and programming tools. A selection of supported expansion boards includes [30]:

- Programming board, including a JTAG programmer and pin headers for all expansion port pins.
- IMU board equipped with gyro and magnetometer.
- Weather station board including sensors for barometric ressure, humidity, temperature, ambient light.
- Gateway board for using a Mulle board as a 6LoWPAN/RPL border gateway.

Fig. 8.5 Current version of a Mulle sensor node (reproduced by permission of Eistec AB, http://www.eistec.se/mulle/wsn)

Mulle runs the open-source Contiki operating system that features a full TCP/IP stack (with support for IPv6) and runs protocols such as UDP, TCP, HTTP and 6LoWPAN among others. The use of TCP/IP over 6LoWPAN enables the Mulle to transmit sensor data directly to the Internet. For Internet operations involving complex negotiations that cause high battery consumption, the gateway acts as a mediator for Internet services, with the aim to reduce the power consumption of the sensors. In its lowest sleep mode, the Mulle's power consumption is 20 μW. Mulle software can be built using the standard GNU GCC tool chain for embedded ARM platforms.

The legacy Mulle has a Renesas M16C/62P microcontroller and either a Bluetooth 2.0 module (v3.1, v3.2, v4.1) or IEEE 802.15.4 transceiver. Similarly as the new version, it also has an on-board 2 MB flash memory.

8.2.6 iSense Core Module 3 (CM30x)

The CM30x [31] uses the 32-bit Jennic JN5148 RISC wireless microcontroller that integrates the radio module in the single chip together with the microcontroller. It has 128 kB of memory that may be shared between program code and data. This flexibility of memory allocation enables a more robust operation than in designs in which the memory allocation is fixed. Its IEEE 802.15.4-compliant radio achieves a data rate of 250 kBit/s and supports two additional modes of operation, offering increased data rates of 500 and 667 kBit/s. The CM30x supports AES encryption, is ZigBee-ready and supports time of flight-based ranging.

There are three variants of CM30x, which differ in the antenna used: the CM30I uses an integrated PCB antenna, the CM30U uses a U.FL connector for external antenna, and the CM30HP uses a power amplifier and U.FL connector for external antenna. The module with the integrated PCB antenna (CM30I) is particularly well

suited for compact systems. For the first two antenna options, a receive sensitivity of −95 dBm (at 250 kBit/s) and a transmit power tunable between −60 and +2.5 dBm is supported while for the third option, a receive sensitivity of −98 dBm (at 250 kBit/s) and a transmit power of up to 10 dBm is supported. The module has a LED that aids in debugging operations and a high-precision clock that with infrequent resynchronization enables precisely timed sleep and wake-up periods. A 34-pin connector that can supply up to 500 mA allows for connection to other modules (such as sensor modules, a gateway module, or an I/O module) to the core module. The core module may be powered by a wall-mount adapter, a standard battery holder, one of the power modules, or via an USB interface. The system has a voltage regulator that is software controlled.

8.2.7 *Fleck3*

The Fleck3 is the second generation of the Fleck series [2, 18], Fig. 8.6. Developed at the CSIRO ICT center in Australia, the Fleck series was designed to overcome some of the limitations of the Mica mote. Areas in which this series overcome the challenges of the Mica2 motes include: the provision of a one-board solution (screw

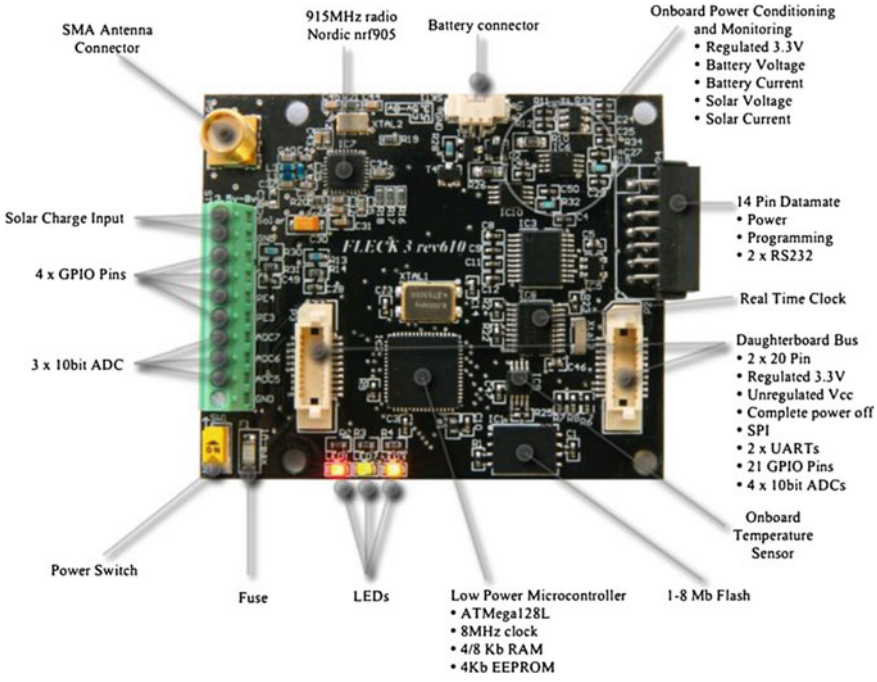

Fig. 8.6 Fleck3 sensor network module (reproduced by permission from [2])

terminals give access to digital and analog channels) and a robust expansion board interface, support for solar charging, and the use of a long-range radio having a range of over 1000 m.

The Fleck3 uses an Atmel ATmega128 microcontroller and a Nordic nRF905 radio. Communication between the radio and the microcontroller is via the SPI bus, for which the ATmega128 acts as the SPI master and the radio as an SPI slave device. The bus is central to the communication mechanism as units such as the flash memory, the real-time clock and the temperature sensor all communicate with the ATmega128 over the SPI bus. The radio works in all the different ISM bands (433, 868, 915 MHz) which is more suited for its main application area (e.g., outdoor environmental monitoring and agriculture applications) than the 2.4 GHz band used by wide-band radios following the IEEE 802.15.4 standard. The radio further has all its components integrated within the chip, which reduces the number of parts and the variability in radio characteristics.

The Fleck3 uses the DS1306 real-time clock (RTC) chip, which is interfaced to the Atmega128 as a SPI slave device. The clock uses <1 μA current for time keeping and <1 mA when active. The RTC frees up the sensor node from the need to keep time, which helps reserve resources for event-based programs. For security, an optional board that uses an off-the-shelf Trusted Platform Computing (TPM) chip from Atmel is supported. This chip implements the full TPM specification that includes a crypto engine, secure storage for keys, and a random number generator. For application development the fleck platform supports TinyOS 1.x.

8.2.8 Cricket

Cricket is a sensor-based device used for indoor localization. It was developed by MIT scientists [17] and manufactured by MEMSIC [26]. It has been widely utilized in research conducted in indoor environments where distance estimation between sensor nodes is needed. There are several advantages of the system including small device size, high measurement accuracy, scalability, user privacy and ease of deployment [26]. A cricket node can be configured as a beacon or a listener. The most common method to use Cricket is to attach the beacons in the infrastructure (walls and/or ceilings) and set listeners on a mobile device that is going to be localized or navigated.

This sensor-based system can provide distance estimates using time-difference-of-arrival (TDoA) with an accuracy of within 1–3 cm. The system utilizes the combination of RF and ultrasound signal to measure distance and provide location information.

The Cricket mote is built on a MICA2 low-power consumption sensor board combined with an RF module and associated hardware to implement the distance measurement feature and the ability for sending distance information. Figures 8.7 and 8.8 show the Cricket hardware implementation graph and the real Cricket mote.

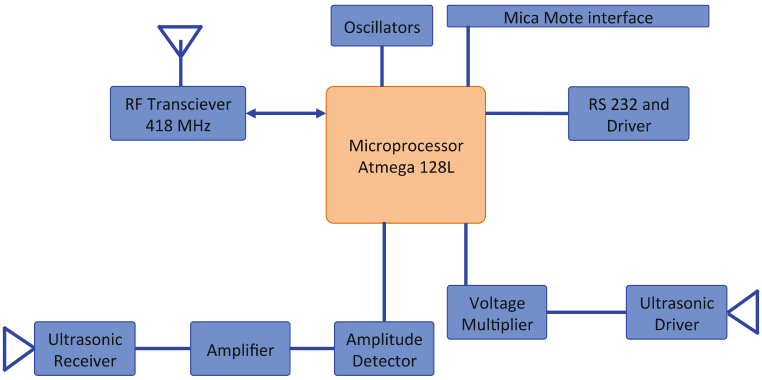

Fig. 8.7 Cricket hardware implementation block diagram with its main components [13]

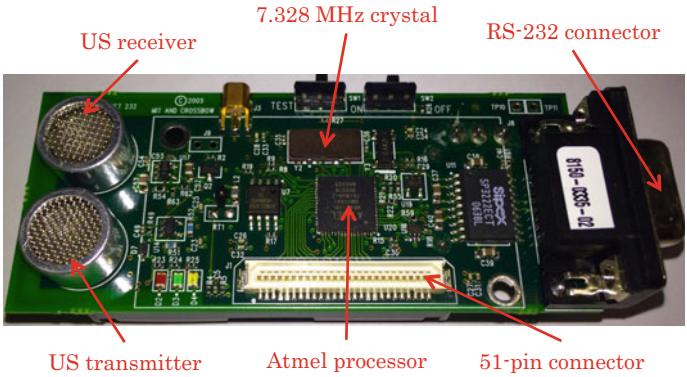

Fig. 8.8 Cricket mote [13]

Cricket mote uses ATmega128L as a main processor, which belongs to the AVR series. It is a high-performance, low-power 8-bit microcontroller that runs at 7.37 MHz in active mode as well as a 32.768 kHz clock that is applied in a power saving mode. The mote also has 128 KB programmable Flash ROM, 4 KB read-only EEPROM, and 53 general-purpose lines. The Cricket uses CC1000 as a low-power RF transceiver, which operates in the frequency range of 300–1000 MHz.

The Cricket system has an ultrasound unit (receiver and transmitter). Within the transmitter, there is a transducer to generate piezoelectric ultrasonic pulses at 40 kHz with configurable duration.

In a typical application using the Cricket system, there are multiple beacons deployed on walls or ceilings that periodically transmit RF and ultrasound signal, Fig. 8.9. At least one listener attached to the host device (laptop, robot) passively receives those signals to calculate the distance between each beacon and itself.

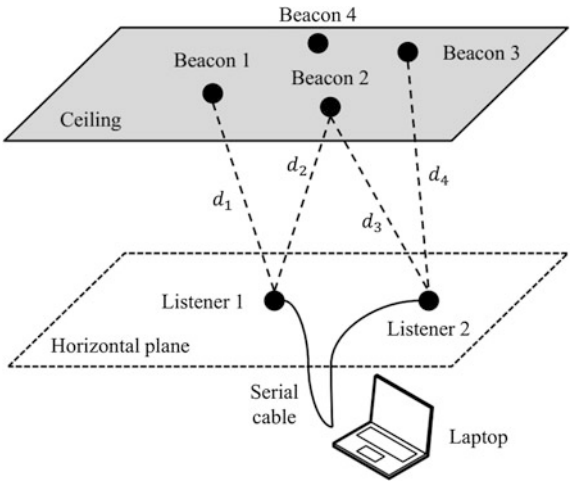

Fig. 8.9 Deployed cricket beacons and listeners in a localization application

The TDoA technique is used to measure the beacon-to-listener distances. The listener estimates distance by obtaining the TDoA of the RF and the ultrasound signals that is given by:

$$\Delta T = \frac{d}{v_{US}} - \frac{d}{v_{RF}}, \tag{8.1}$$

where d is the distance from the beacon to the listener, v_{US} is the speed of sound propagation (in a normal room temperature and humidity $v_{US} \cong 344$ m/s) and v_{RF} is the speed of light ($v_{RF} \cong 3 \times 10^8$ m/s). With $v_{RF} > > v_{US}$, the distance is given by an approximation: $d \approx \Delta T \times v_{US}$.

8.2.9 *Shimmer Wireless Node*

Developed by Shimmer Research™, the Shimmer Wireless Platform was designed from the ground up to be a low-power, modular, wearable sensor node for use in healthcare, sports science, environmental sensing, and biofeedback [25]. A typical platform would include the main unit and a sensor board. The packaging is designed so that the main board rests at the bottom of the package and the sensor board rests on the top.

The main node consists of a Texas Instruments MSP430F1611 microcontroller featuring an 8 MHz clock, 2 DAC outputs, and an 8-channel, 12-bit ADC. The microcontroller also includes 10 KB of RAM and 48 KB of flash memory. Two sensors are integrated onto the main node (three-axis accelerometer and a tilt/vibration switch). The Shimmer Wireless Platform uses dual radios including TI CC2420 radio chip for 2.4 GHz IEEE 802.15.4 communication and the Bluetooth

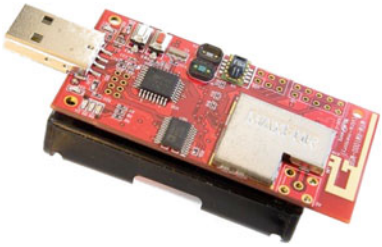

Fig. 8.10 XM 1000 wireless sensor node (reproduced by permission of ADVANTICSYS, [32])

radio. Power is provided to the node with a Li-ion rechargeable battery. The node will accept voltages from 2.2 to 3.6 V.

Some examples of sensor modules include the 9DoF kinematic, GPS, and ECG. The 9DoF kinematic sensor board features a Honeywell HMC5843 three-axis digital compass and an InvenSense 500 series MEMS-based gyro. These sensors allow for complex motion sensing in a 3D environment. Worn on a user, it enables gestural computing and can be used for virtual reality and gaming applications.

8.2.10 ADVANTICSYS XM1000

The ADVANTICSYS XM1000 is a wireless sensor node based on an upgraded Crossbow TelosB platform [32] and is powered by two AA batteries, Fig. 8.10. It features the 16 MHz, 16-bit TI MSP430F2618 microcontroller using the RISC instruction set. Integrated into the microcontroller are 116 KB of program flash and 8 KB of data RAM with node-integrated external flash chip. The expanded memory allows for over-the-air reprogramming of the device as well as implementation of Device Profile for Web Services (DPWS)—a method for web service discovery and communication with the device on a LAN or WAN. The microcontroller enables communication using UART, SPI, and I^2C protocols.

Several sensors are directly integrated onto the board for monitoring temperature, humidity, and light. Wireless connectivity at 2.4 GHz is enabled by the TI CC2420 transceiver. This chip is IEEE 802.15.4 compliant, and has a range of up to 120 m outdoors and 30 m indoors.

8.3 WSN Simulation Tools

When conducting research on WSNs, it is often more feasible to use WSN simulations than deploy a real WSN. One common reason for preferring a WSN simulation is that of minimizing costs—for example, the total cost of purchasing WSN nodes and other hardware would be prohibitive for applications requiring very large number (e.g., thousands) of nodes. In such cases, researchers use WSN simulations,

with the decision to deploy the real nodes depending on the promise depicted by the simulation-based research. Cost limitations are however not the only reason in support of WSN simulations. Other reasons include [5]: (1) *Network Debugging Issues*—debugging large distributed networks can sometimes be a daunting task and simulation can, for certain scenarios, provide a means to find and correct bugs prior to undertaking real WSN implementations. (2) *Harsh Operating Environments*—the target environments for certain WSN applications are unsafe for humans. Examples of applications associated with such environments include those for monitoring wildlife, volcanic activity or adverse weather conditions to mention but a few. In such cases, simulation offers a means to conduct experiments in a safe environment before resources and necessary mitigations against threats can be dedicated to deployments in the real operation environment. (3) *The Need for Precise Control of WSN Parameters*—in WSN research, it is often required to evaluate the behavior of the network for different combinations of precisely chosen parameter settings. Because a live WSN setting is not entirely controllable by the experimenter, simulations come in handy to provide this controlled and repeatable environment.

To address challenges such as these, the community has put forth a number of simulators. This section briefly explores five of the most popular WSN simulation tools.

8.3.1 ns-2 (Network Simulator-2)

Network Simulator-2 (ns-2) is an open-source discrete-event simulator. The simulation kernel, models and protocols are implemented in C++ while the creation, control and management of simulations is done in Object-oriented Tool Command Language (OTcl) [5, 8]. Network Simulator-2 was designed for simulating traditional IP networks and as such requires special extensions in order to support WSN simulations. The extensions augment the base ns-2 functionality with features such as sensing, processing energy consumption, WSN operation modes (e.g., sleep and wake-up modes), variations in node capabilities (e.g., regular nodes versus access points), and various options for dissemination of sensed data among others. Two of these extensions such as MannaSim [14] and SensorSim [33] have been widely used in past WSN research. SensorSim has however long ceased to be developed or supported and we do not discuss it here.

MannaSim comprises two components: (1) the MannaSim Framework, which encompasses the core extension module used for the design, development and analysis of different WSN applications; and (2) the Script Generator tool, which provides a front-end via which Tool Command Language (Tcl) simulation scripts are easily created. MannaSim enables the user to control aspects of the network's composition (e.g., number, type and density of nodes) and its organization (e.g., flat or hierarchical network). It supports a wide range of applications and provides a testbed for various algorithms and protocols [14]. MannaSim comprises a set of

classes, which extend the corresponding ns-2 classes. For example, the *Battery* class extends ns-2's *EnergyModel* class and provides methods for the implementation of various battery models. Another example is the *SensorNode* class, which extends ns-2's *MobileNode* class by adding features such as sensing and processing. A full list of classes can be found in [14, 20].

One of the major advantages of ns-2 as a WSN simulator is its abundance of publicly available protocols and algorithms [22]. Some of its most notable weaknesses on the other hand include the steep learning curve that one typically goes through before undertaking meaningful simulations and the absence of an application model [22].

8.3.2 OMNETT++

OMNETT++ is a discrete-event simulation environment for communication networks in general. Its components are developed in C++ while the simulation implementation is based on a high level language called Network Description Language (NED) [20]. Because OMNETT++ is not specifically designed for WSNs, it requires special packages to be able to run WSN simulations. Below, we briefly discuss two of the most popular amongst these frameworks.

Castalia: Via a wide range of tunable parameters, Castalia can be used to simulate a broad spectrum of WSN platforms. Its most outstanding features include [34]: (i) the advanced channel model which is based on empirically measured data; (ii) the radio model which is based on real radio components; (iii) its extended modeling sensing provisions, and (iv) its intrinsic design for adaptability and extensibility. The latter attribute in particular enables researchers to easily port their algorithms and protocols into Castalia. Castalia is not recommended for cases where emphasis of simulation is to observe detailed platform-specific behavior. In such cases, Castalia is best used as a first-line simulator that provides a coarse-grained view of the WSN's behavior before more fine-grained platform-specific simulations can be run [34].

MiXiM (*Mixed Simulator*): It is a merger of several OMNETT++ based frameworks for mobile and wireless simulation [35]. The word "mixed" in its name comes from its being a combination of various simulators [20]. Its radio propagation model is based on the Channel Simulator (ChSim) [36], connection management and mobility support are based on the Mobility Framework (MF) [37] while the protocol library is derived from the MAC simulator [38], the Positif *Framework* [38] and from the *Mobility Framework*. MiXiM supports models based on both 2D and 3D settings and, in addition to traditional WSN nodes, supports the simulation of objects such as walls and houses as obstacles to the propagation of radio waves.

8.3.3 TinyOS Simulator (TOSSIM)

TOSSIM is a discrete-event simulator specifically designed for TinyOS applications. It has support for two programming interfaces, one of which is in Python, the other in C++. Using the Python interface, one can interact with the running simulation dynamically. TOSSIM programs can replace entire components of TinyOS applications with their simulation implementations since these programs in general require no modification to be run on the motes [1, 28]. This feature gives developers a big margin for application testing and debugging since TinyOS applications may be developed and compiled to the TOSSIM framework running on a desktop. TOSSIM captures details of TinyOS's behavior, closely simulating each ADC capture and each interrupt to the system. Using tools external to TOSSIM, users can implement models of different real-world phenomena.

The design choice to keep real-world models external to the simulator was aimed to allow the flexibility for researchers to implement their own models in such a way that the TOSSIM environment imposes no definitions of its own regarding what is correct or wrong [11]. Despite capturing TinyOS behavior at a very fine level, it is noteworthy that TOSSIM makes several simplifying assumptions that could cause unexpected behavior. For example, as a direct consequence of being a discrete-event simulator, TOSSIM interrupts are non pre-emptive. If, as an example, pre-emption were to put a real-world mote into unrecoverable state, an equivalent simulated TOSSIM mote would not capture this behavior [11]. These kinds of challenges notwithstanding, TOSSIM offers a good first step towards the understanding of algorithm performance prior to implementation on a real WSN.

8.3.4 Optimized Network Engineering Tool (OPNET)

OPNET is a proprietary[1] discrete-event simulator targeted for computer networks in general. Different from simulators such as ns-2, OPNET supports the modeling of sensor hardware (e.g., transceivers and antennas), and has provision for the definition of custom packet formats [1]. Using its GUI, one may model, graph or animate the simulator's output. Researchers have designed an interface for compiling TinyOS applications to OPNET models [21]. Just as the kind of interoperability realizable between TOSSIM and TinyOS, this interface enables a shared code model in which the same application code is shared between the TinyOS executable and the OPNET simulation model. The interface has support for scenario management and statistics management, has the ability for instantiations of different applications to be simulated together in the same memory space, and has the availability of a much larger wealth of models, the combination of which give OPNET an edge when it comes to sophisticated TinyOS simulations [21].

[1]Since October 2012, the simulator is owned by Riverbed [40].

Typical of proprietary products, the OPNET license comes with a considerable amount of documentation and study cases that can be useful to researchers [1].

8.3.5 Avrora

Avrora is a Java-based open-source simulator for embedded sensing programs. It simulates the actual microcontroller programs (as opposed to models of the software), and executes cycle-accurate simulations of the devices and the radio communication [23]. It can scale to networks of up to 10,000 nodes and can handle as many as 25 nodes in real-time [23]. It has a nearly complete implementation of the mica2 hardware platform, an ATMega128L implementation, and an implementation of the CC1000 AM radio [39]. To enable testing, debugging, or analyzing of programs before running them in network simulations, Avrora supports the simulation of a sensor network program on a single node. For complete network simulation Avrora provides full timing accuracy and allows programs to communicate via the CC1000 AM radio. Two notable challenges seen with Avrora are that it is slow (e.g., it is 50 % slower than TOSSIM) and it does not model node mobility or clock drift [22].

References

1. A. Abuarqoub, F. Al-Fayez, T. Alsboui, M. Hammoudeh, and A. Nisbe, "Simulation issues in wireless sensor networks: a survey," *Proc. the Sixth International Conference on Sensor Technologies and Applications (SENSORCOMM 2012)*, Rome, Italy, 2012.
2. P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore, "Environmental wireless sensor networks," *Proceedings of the IEEE, Special Issue on Emerging Sensor Network Applications*, vol. 98, no. 11, pp. 1903–1917, November 2010.
3. *Cricket User Manual*, MIT Computer Science and Artificial Intelligence Lab, Second Edition, Jan. 2005.
4. C.-M. Hsieh, F. Samie, M.S. Srouji, M. Wang, Z. Wang, and J. Henkel, "Hardware/software co-design for a wireless sensor network platform," *2014 International Conference on Hardware/Software Codesign and System Synthesis (CODES + ISSS)*, New Delhi, 2014.
5. M. Jevtic, N. Zogovic, and G. Dimic, Evaluation of Wireless Sensor Network Simulators, *17th Telecommunications forum TELFOR*, Belgrade, Serbia, 2009.
6. M. Johnson, M. Healy, P. van de Ven, M.J. Hayes, J. Nelson, T. Newe, and E. Lewis, "A comparative review of wireless sensor network mote technologies," *Proc. IEEE Sensors*, pp. 1439–1442, October 2009.
7. S.D.T. Kelly, N.K. Suryadevara, and S.C. Mukhopadhyay, "Towards the Implementation of IoT for Environment Condition Monitoring in Homes," *IEEE Sensors Journal*, vol. 13, no. 10, October 2013.
8. S. Khan, A.K. Pathan, and N.A. Alrajeh (Editors), *Wireless Sensor Networks: Current Status and Future Trends*, CRC Press, Taylor & Francis Group, Boca Raton, Florida, 2013.
9. J. Ko, K. Klues, C. Richter, W. Hofer, B. Kusy, M. Bruenig, T. Schmid, Q. Wang, P. Dutta and A. Terzis, "Low power or high performance? A tradeoff whose time has come (and nearly gone)," *Proc. the 9th European Conference on Wireless Sensor Networks,* Italy 2012.

10. M. Kohvakka, T. Arpinen, M. Haunikainen and T.D. Hamalainen, "High-performance multi-radio wsn platform," *Proceedings of the 2nd International Workshop on Multi-hop Ad Hoc Networks: From Theory to Reality*, pp. 95–97, 2006.
11. P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," *Proc. of the 1st International Conference on Embedded Networked Sensor Systems*, New York, NY, USA, 2003.
12. A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," *Proc. the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA '02)*, pp. 88–97, 2002.
13. MIT Computer Science and Artificial Intelligence Lab, "Cricket user Manual, Second Edition," January 2005.
14. R.M. Pereira, L.B. Ruiz, and M.L.A. Ghizoni, "MannaSim: A NS-2 Extension to Simulate Wireless Sensor Networks," The Fourteenth International Conference on Networks, Barcelona, Spain, 2015.
15. F. Philipp, F. A. Samman and M. Glesner, "Design of an autonomous platform for distributed sensing-actuating systems", *22nd IEEE International Symposium on Rapid System Prototyping (RSP)*, pp. 85–90, 2011.
16. J. Portilla, T. Riesgo and A. De Castro, "A reconfigurable FPGA-based architecture for modular nodes in wireless sensor net-works," *3rd Southern Conference on Programmable Logic*, Mar del Plata, Argentina, 2007.
17. N.B. Priyantha, *The Cricket Indoor Location System*, Ph.D. Dissertation, Massachusetts Institute of Technology, Jun. 2005.
18. P. Sikka, P. Corke, L. Overs, P. Valencia, and T. Wark, "Fleck—a platform for real-world outdoor sensor networks," *Proc. 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, Melbourne, Australia, pp. 709–714, December 2007.
19. G. Stamatescu, C. Chiţu, C. Vasile, I. Stamatescu, D. Popescu and V. Sgârciu, "Analytical and experimental sensor node energy modeling in ambient monitoring," *9th IEEE Conference on* Industrial Electronics and Applications (ICIEA), Hangzhou, China, 2014.
20. M. Stehlik, *Comparison of Simulators for Wireless Sensor Networks*, Master Thesis, Masaryk University, 2011.
21. D. Sumorok, D. Starobinski, and A. Trachtenberg, "Simulation of TinyOS Wireless Sensor Networks Using OPNET," *Proc. of OPNETWORK 04*, Washington DC, August 2004.
22. H. Sundani, H. Li, V.K. Devabhaktuni, M. Alam, and P. Bhattacharya, "Wireless sensor network simulators, a survey and comparisons," *International Journal of Computer Networks (IJCN)*, vol. 2, no. 5, 2015.
23. B.L. Titzer, D.K. Lee, and J. Palsberg, "Avrora: scalable sensor network simulation with precise timing," *Proc. the Fourth International Symposium on Information Processing in Sensor Networks*, UCLA, Los Angeles, CA, 2005.
24. J.S.C. Turner, M.F. Ramli, L.M. Kamarudin, A. Zakaria, A.Y.M. Shakaff, D.L. Ndzi, C.M. Nor, N. Hassan, and S.M Mamduh, "The study of human movement effect on signal strength for indoor WSN deployment," *IEEE Conference on Wireless Sensors (ICWiSe2013)*, Kuching, Sarawak, December 2013.
25. "Shimmer Wireless Sensor Unit/Platform." [Online]. Available: http://www.shimmer-research.com/p/products/sensor-units-and-modules/shimmer-wireless-sensor-unitplatform.
26. http://www.memsic.com/wireless-sensor-networks/.
27. http://www.digi.com/lp/xbee.
28. http://www.tinyos.net/.
29. http://wsn.cse.wustl.edu/images/e/e3/Imote2_Datasheet.pdf.
30. http://www.eistec.se/mulle/.
31. http://www.coalesenses.com.
32. http://www.advanticsys.com.
33. http://www.nrl.navy.mil/itd/ncs/products/sensorsim.
34. https://castalia.forge.nicta.com.au/index.php/en/.
35. http://mixim.sourceforge.net/.

36. http://www-old.cs.uni-paderborn.de/en/fachgebiete/research-group-computer-networks/projects/chsim.html.
37. http://mobility-fw.sourceforge.net/.
38. http://www.consensus.tudelft.nl/software.html.
39. http://compilers.cs.ucla.edu/avrora/sensors.html.
40. http://www.riverbed.com.