

was responsible for supplying hardware for testing. Currently Crossbow is a producer of tiny motes.

Since Culler's (The chairmen of Smart Dust program) team and cooperators have offered the Mote hardware and the TinyOS and TinyDB software as open source, researchers worldwide on it have jumped on. Following professor Culler more than 100 teams around the world are using the combination of open-source Motes with the TinyOS [operating system] and TinyDB [database].

Smart Dust has potentially widely usage not only for obvious military propose but for civilian as well. For example environmental monitoring, building construction monitoring, forest fire detection, flood detection, weather forecast or usage like Ivy program. The goal of Ivy is to provide a research infrastructure of networked sensors for the College of Engineering at UC Berkeley.

Such kind of new technology caused of course a lot of problems to resolve, efficiently power supplier, „smart” protocols and algorithms which would be able to work correctly in a shortage of memory and processor capacity. One of research work is focused on nanotechnology, which seems to be promising perspective of resolving hardware problems. A lot of research centers is working on it. For example in Germany at Max Plank Institute where professor Herbert Walther with co-workers is working on isolating and grouping single atoms in order to achieve computer system or doctor Isaak Huang from IBM (Silicon Valley) laboratory where they learn compute a fluorine solution to expose it to magnetic field and microwaves.

2. Hardware

As was mentioned above two companies cooperate with Berkeley and were involved in hardware development Intel and Crossbow. That companies offered mote devices on market.

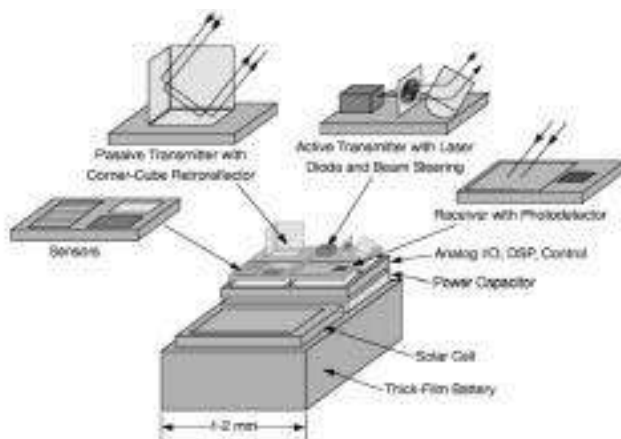


Fig. 1. Smart dust mote, containing micro fabricated sensors, optical receiver, passive and active optical transmitters, signal-processing and control circuitry, and power sources.

First reviewing of needed hardware technology in Smart Dust was presented in [1]. The possibility of introducing smart dust was caused by rapid convergence of three technologies: digital circuitry, wireless communication and Micro Electro-Mechanical Systems. The requirements toward Motes are as follow: as small as possible, extremely low energy consumption, ability to sensing and communicate to each other, acting independently, low cost, multifunctional sensing, wireless communication and computing ability.

Above scheme [1] presented idea of Smart Dust integrated single small package contained all needed components for communication and sensing with own power supply. It used Optical Transmission as communication equipment. One of the main challenges of it is performing all needed functions with low power consumption. Taking into consideration that battery in size of AA have 1 J of energy means that one day working motes could consume not more than 10 μ W. So effort of device designers is focused on constructing low consuming components as small as possible and the aim for the software designers is to produce efficient low power consuming software.

Currently existed motes characteristic example on table 1.






Table 1. Example of motes characteristic

Dimensions	3 cm x 3 cm
CPU Speed	4 MHz
Memory ROM	128 Kb FLASH
Power supply	2X AA batteries
Power consumption	Power consumption
Processor	5.5 mA (active mode) <20 μ A (sleep mode)
Radio	12 mA (transmit mode) 1.8 mA (receive mode) <1 μ A (sleep mode)
Output Device Network	3 LEDs Wireless 4 Kb/s at 916 MHz

The next problem to resolve is how to allocate energy the for Sensing, Computing and Transmitting. There are two main candidate for communication radio frequency (RF) and Optical Transmission (OT). Both techniques have advantages and disadvantages. RF requires antenna, radio transceiver is relatively complex and required more energy. Optical transmissions need less energy does not require antenna, what make mote smaller but communication between randomly deployed nodes is hardly possible. In such case only communication base station – node is possible what constrains the scope of optical transmission. It also requires directive transmissions and produced many problems connected with laser beam characteristic (table 2).

Two mote hardware platforms, the rene2 and the mica, are currently available for general development. Rene2 is older one and is not implemented in software platform designed and published as a open source TinyOS. There are two implemented and simulated platform mica and mica 2.

Table 2. The family of Berkeley TinyOS motes [3]

Mote Type	weC	rene2	rene2	dot	nica
					
Date	9/99	10/00	6/01	8/01	2/02
Microcontroller					
Type	AT90LS8535		ATMega163		ATMega103
Prog. mem. (KB)	8		16		128
RAM (KB)	0.5		1		4
Nonvolatile storage					
Chip	24LC256			AT45DB041B	
Connection type	I2C			SPI	
Size (KB)	32			512	
Default Power source					
Type	Li	Alk		Li	Alk
Size	CR2450	2xAAA		CR2032	2xAAA
Capacity (mAh)	575	2850		225	2850
Communication					
Radio	RFM TR1000				
Rate (Kbps)	10	10	10	10	10/40
Modulation type	OOK				OOK/ASK

3. Software

3.1. Positioning

One of the most important issue in Smart Dust is problem how to establish position and conneted with it how to measure distance between the nodes. In self organised network Nodes used own algorims for establishing the nearest neighbourhood and recognizing the best routing path to the base station. Knowledge about its own position is crucial. There have been many attempts to solve the ad-hoc network localization problem [4], [5], [6], [7].

Correct position measuring depends on:

- first, analyzing the hardware for predictable characteristics on which computational solutions can be built;
- second, relying on well-founded mathematical optimization theories when designing computational solutions.

Kamin Whitehouse David Culler in their work „Calibration as Parameter Estimation in Sensor Networks” [5] propose ad-hoc localization system for sensor networks they parameterize each device and choose the values of parameters that optimize system performance. Such method could reduce average error from 74% to 10% additionally they propose to expand this technique to a method of auto-calibration for localization problem. They proposed ad-hoc localization system called CALAMARI. Calamari estimates distance between nodes counting the difference in time reaching between radio signal RF (Radio Frequency) and acoustic signal TOF (Time of Flight). One node transmits RSSI (Radio Strength Signal) 916.5 MHz and another samples it. An acoustic pulse

of is transmitted by 15 ms along with a 25 ms radio packet. Because nodes are equipped with radio for communication this is cheap metered. TOF requires special hardware but is better in estimation distance than radio. The transmitter sends radio and acoustic signal in the same time. The receiver measures the difference in arrival of both signals. Radio and sound travel at different speeds, difference of arrival shows distance between them. Cheap mass produced nodes may provide different radio strength signal and different frequency what caused wrong distance estimation. The acoustic inaccuracy is the same. Above factor in two different transmitter-receiver pairs may vary even to 300%.

Calamari methods of calibration ad-hoc sensor networks has three steps:

- parameterize each individual device and model the system response as a whole using these parameters,
- collect data from the system as a whole,
- choose the parameters for the individual devices such that the behavior of the entire system is optimized.

This technique of choosing the parameters for individual devices are proceed by observing *trends* in the transmitter/receiver pairs. In other words, if all distance estimates made with a particular transmitter are high the transmitter is wrong. Proceeding this way is able to assess the node [2].

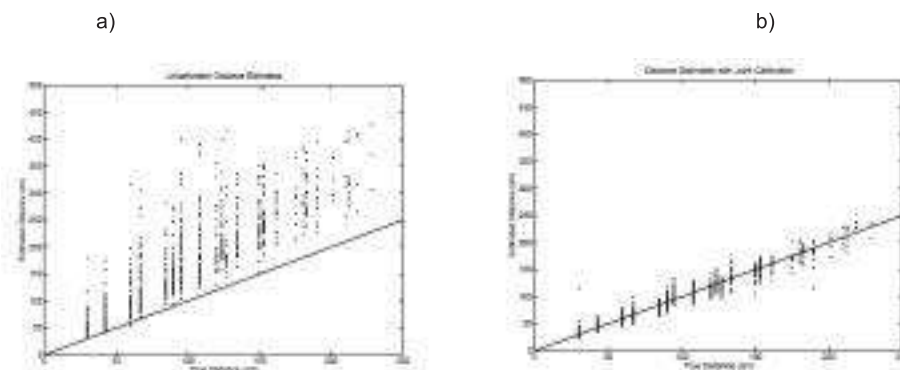


Fig. 2. Calamari estimation [3]: a) before, b) after process

Calamari authors propose auto calibration. In ad-hoc node network every transmitter/receiver pair is also receiver/transmitter so there are two distance estimations for each pair of nodes. First assumption to estimate the parameters is to select those, which maximize the consistency of the responses between the symmetric pair. Second assumption is that all distance estimates must satisfy the triangle inequality. Two-assumption together gives constrained optimization maximizes consistency and satisfying the triangle inequality.

Calamari method shows that macro-calibration is possible and is more efficient than micro-calibration and the second advantage is ability to calibrate sensors in different environment with knowing arbitrary parameter estimation technique.

The second approach to localization problem is trying to correct positioning accuracy used mathematic model derived from experimental observing [4]. Xuan Long Nguyen & Tye Rattentbury in their work „Localization algorithms for sensor networks using RF signal strength” presented algorithm, which after calibration - established position of each node in network.

They experimentally established that, there is a relation between distance from the center of network and estimation error. Observed that the position given by bounding box method are always pulled towards center of the network, so the longer distances nodes are underestimated and the shorter distance are overestimated. As a result of that observation the new function of positioning was introduced which takes into consideration distance from the center (it was incorporates as a hidden error).

Algorithm for establishing the position:

1. Randomly generate nodes inside of a rectangular region.
2. Randomly connect nodes iid with probability p
3. Randomly choose initial anchor nodes iid with probability q
4. Randomly choose line parameters that satisfy: slope < 1 and intercept > 0 .
5. For connected nodes, set the distance estimations between nodes

Diagrams on fig. 3 show estimation errors before and after introduced algorithm with hidden error.

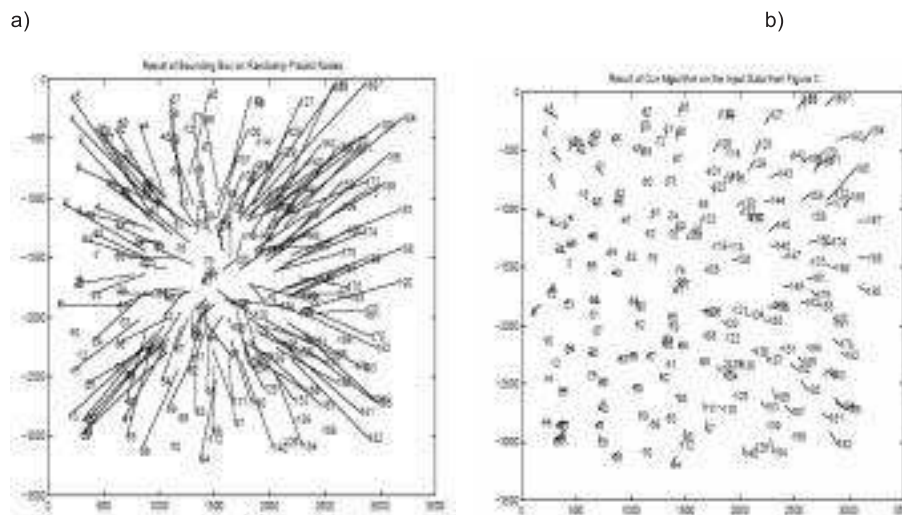


Fig. 3. Estimation errors before and after introduced algorithm with hidden error;
a) before introduction algorithm, b) after introduction algorithm.

3.2. Routing

Routing is connected with a problem of proper choosing the path to the station. There are two types of networks routing:

- single hop: where all devices are in the same scope so the data between node and base station are sent only by one hope.
- multi hop: where data are not in the same scope and another intermediate nodes take part in data sending.

The multi hop algorithms are of course much more complicated and there is a higher possibility of losing data, searching of new path, retransmitting and so on.

There is second factor, which divides routing protocols algorithms:

- proactive: traditionally used in computer networks. Single node in proactive routing algorithm keeps tracking to all nodes in the network even to those where it newer sent data. It require serial renewing information about path to each node and it produce relatively big movement in network to support the routing table in each node.
- active: protocols which show the path to the next node only for demanding.

There are some difference between Sensor Network and ad-hoc Sensor Network. In Sensor Network there are same specialization and some node could be responsible for sensing some for routing. In ad-hoc Sensor Network any node should be able to support routing between any node.

There are many routing protocols designed for ad-hoc Sensor Network some of them are described below.

- TinyOS beaconing [8] - constructs routing tree at the base station and periodically updates each node information from base station. Every node receiving routing updating considers the node from which it received information as a first candidate for forwarding packet.
- Direct Diffusion [8] is a data centric protocol. The Base Station setting up gradients with the network and sending requests to node to obtain data it need. The node possessing interesting information sends it back the same path to the Base Station.
- Geographic Routing [8] is a routing leverage to the node positions and routed packets to the node, which is closest to the destination
- Minimum cost forwarding [8] is a protocol where Base Station counting the cost for obtaining information. As a cost may be taken: hops, energy, latency and so on
- LAR (Location Aided Routing). Its acts as proactive protocols - demanding of path are announced into the network but it sends it only to nodes situated on the way to the station. It established so called „forwarding zone” and send packet only toward the zone.
- GRID is a routing protocol, which divided area into two-dimensional scope. In each rectangle (described by grid) one node is chosen as a leader and this node is responsible for packet forwarding. The other nodes situated in

rectangle do not take a part in packet forwarding process. If any wanted to send data communicate itself to the leader of own rectangle. If leader leave rectangle the other node is chosen as a leader. Similarly to the LAR protocol the information about position is used for distinction forwarding zone, but in GRID only the leaders of rectangle take part in it. The path is determine not by node but by rectangle and even if node used to forwarding packet leave rectangle or stop working the other node is able to overtake it function. This feature made GRID protocol insensitive for movement of node or breakdown some of them.

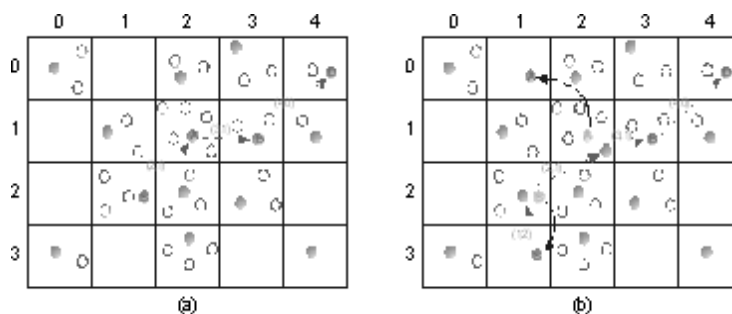


Fig. 4. Algorithm of acting GRID routing introduction

3.3. Security

One of the important item in ad-hoc sensor network is reliability and connected with it security. The problem of security in ad-hoc sensor networks is presented in [8]. As was mentioned above the biggest problem with Smart Dust is power consumption. In traditional network most security checks taking place in higher layer. In ad-hoc sensor network packet forwarding is executed by end-to-end algorithm so each node has access to the proceeding data and even aggregate it to constrain network traffic. So enemy attack to such designed network isn't problematic. Specially attack by using bogus routing information. The authors' [8] suggested than sensor network routing protocols must be designed with security in mind. There are many ways to attack ad-hoc sensor network. First attacker used own bogus motes which acting like the other original and spoofing data passed by network, it could also generate errors message, or drop forwarding packet down, it may also changing the original packet and replace it by own.

Other kind of attack is so called „sink hole”- the compromised node in respective to routing algorithm is particularly attractive for surrounding nodes and all traffic with it neighborhood is send by it. Such bogus node enables any above described attack. The next potential attack is „Sybil attack” in which single node present multiple identity what may disorganize network by sending different information each time for example about own position. The other possible attack for ad-hoc sensor network is to generate high traffic what cased

discharging of node energy. There are many others ways of attack but described above seem to be the major.

The next problem is how to protect the nodes from the attack. What is the remedy. One of the way is layer encryption. Encrypted packets should be resistant to such attack like Sybil, selective forwarding but not for wormhole or insider attack. In work [8] authors propose many remedies how to fight against knowing attacks for example: multi path routing, identity verification, and bi-directional link verification, authenticated broadcast but all of them require processor capacity and consumes power energy.

Interesting proposal is included in study [9]. Author's worked out hardware for cryptography based on Hashed Functions for authentication. MAC – Message Authentication Codes are useful tools for ensuring the integrity of data. The main target was providing universal hash function for ensuring security in sensor network communication. They worked out three hashing functions. PH – which produces hash of length $2w$, PR and WH produces for much shorter hash length than w . Following their opinion they achieved 59% power savings and speed up process of hashing 7.4 times compare to other hashing functions what made it useful in ultra low power sensor like Smart Dust.

3.4. Operating System and Tools

The most popular operation system is designed by Berkeley University system named „TinyOS“. A lot of research centers use it because it is allowed for using as a Open Source. TinyOS [3] is a operation system designed specially for sensor networks. It is written in nesC language, which was implemented specially for this operating system. It is composed from modules of software components. On the components level it contains three abstractions:

- commands there are used for component (for example telling the network component to send the packet)
- events are calls up the component (for example signaling the packet has been sent)
- tasks are used for computation event or command demanding.

TinyOS schedules tasks on a FIFO basis and runs a task to completion before another task is run. Otherwise, the task queue can overflow as new tasks are posted. TinyOS supports high concurrency through split-phase non-blocking execution. TinyOS is an asynchronous system. TinyOS provides high parallelism and efficiency thanks to programming interface. For network communication TinyOS use high level abstraction Active Message (AM). AM is a unreliable data link protocol. Taking into consideration the limitation of Smart Dust devices there are some rule require during projecting process so application should be: small, expressive, concise, resilient, efficient, tailorable, simple.

In study [3] authors for improving efficiency of using the TinyOS propose virtual machine named „Mate“. Mate is a small environment consists only 24 instructions in which is possible to built all needed component running under TinyOS system. Mate is resilient to failure it provides network and sensor ac-

cess and can quickly implement algorithms. It seems to be easy and productive tool for protection of hardware abstraction from application code in sensor network. Authors declared their future work on it to provide application specific virtual machines and the user-land abstraction.

3.4.1. The NesC language

The grammar of nesC is an extension the ANSI C grammar [10]. A nesC application, as a tool for TinyOS implementation consists of one or more components linked together to form an executable program. A component provides and uses Interfaces. Interfaces are bi-directional access points to the component. An interface provides commands - a set of functions, which the interface provider must implement, and events - a set of functions, which the interface user must implement. For a component to call the commands and events must be implemented. Typically commands call downwards, while events call upwards. A single component may use or provide many interfaces and instances of the same interface.

There are two types of components in nesC: modules and configurations. Modules provide application code and implementing one or more interface. Configurations are used to link components together, it connects interfaces used by components to interfaces provided by others. This is called wiring. Components have internal concurrency in the form of tasks, control of concurrency system recognizing as a thread is passed into component by its interface. The concurrency model of nesC is based on run-to-completion tasks, and interrupt handlers which may interrupt tasks and each other. Every nesC application is described by a top-level configuration that wires together the components inside.

3.4.2. TOSSIM

TOSSIM, the TinyOS discrete simulator [11], compiles directly from TinyOS code. Instead of compiling TinyOS application for a mote, for example „make mica”, user may build TOSSIM simulation with „make pc”, in Linux environment. The simulation may run natively on a desktop or laptop. TOSSIM can simulate thousands of nodes simultaneously. Every mote in a simulation runs the same TinyOS program. TOSSIM provides run-time configurable debugging output, allowing a user to examine the execution of an application from different perspectives without recompiling source program. Debugging may be declared as an environmental set in Linux operating system or declared during program writing. TOSSIM of course is only a simulation environment and like each modeling it makes several simplifying assumptions. In study [11] they are enumerated.

- Fidelity - TOSSIM captures TinyOS' behavior at a very low level. It simulates the network at the bit level, simulates each individual ADC capture, and every interrupt in the system.

- Time is simulating, as 4 MHz CPU but it does not model execution time.
- Models – it provide the abstraction of real world phenomena which may be implemented by user following his/her needs.
- Building- it derives directly from TinyOS code. So to build TOSSIM first TinyOS program should be done and afterward by using „make pc” command-running simulation on PC is available.
- Imperfections - TOSSIM captures TinyOS behavior at a very low level but it makes several simplifying assumptions. So it is possible that code which runs in a simulation, on a real mote dos not work.
- Networking - TOSSIM simulates the 40Kbit RFM „mica” networking stack.

3.4.3. Visualization Toll

One of the most common used visualization tools in Linux environment is TinyViz. TinyViz is a Java-based GUI that allows visualizing and controlling the simulation as it runs. It inspecting debugs messages, radio and UART packets, and so forth. The simulation provides several mechanisms for interacting with the network. For example packet traffic can be monitored, packets can be statically or dynamically injected into the network and so on. TinyViz is not actually a visualize, it is a framework in which plug ins can provide desired functionality. It posses interface in which user can handle with.

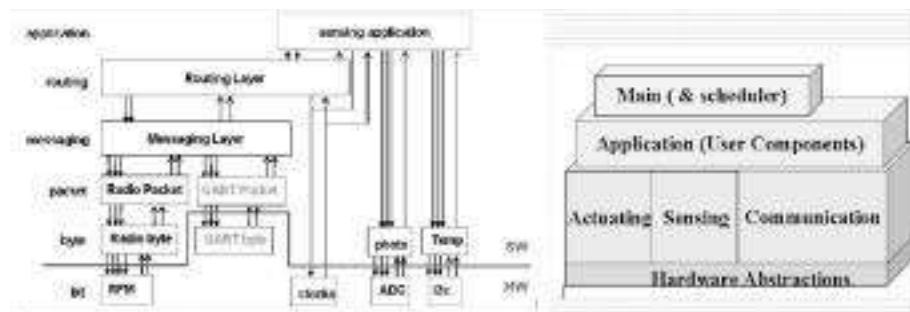


Fig. 5. TinyOS [13]

4. Summary

Presented above items shortly described problems and developing trends connected with Smart Dust issue. Although this way of developing computer research seems to be promising it also creates a lot of new issues to resolve. Frankly speaking no one is able to predict if this direction of searching bring real breakthrough. So far dealing with ad-hoc sensor network bring same successes but creates more problems to resolve. Following authors opinion the

promising direction is nanotechnology but without resolving the problem of efficient power supply the real progress is impossible.

The second issue for developing in this field is efficient self-organizing of nodes. Although the algorithms presented above (communications, routing, positioning) are some proposals but it is a lot have to do in this field. The third issue there are problems which are not taken in to consideration so far, for example density of nodes deploying in depend on distance from the base station.

Literature

- [1] Joseph M. Kahn, Randy Howard Katz, and Kristofer S. J. Pister Emerging Challenges: Mobile Networking for "Smart Dust". (<http://www-ee.stanford.edu/~jmk/pubs/jcn.00.pdf>)
- [2] Sotiris Nikolettseas: Efficient Data Propagation Algorithms in Smart Dust Networks. (<http://www.cs.cas.cz/sofsem/06/data/prezentace/22/nikolettseas.pdf>)
- [3] Philip Levis, David Culler Mate: A Tiny Virtual Machine for Sensor Networks (<http://www.cs.berkeley.edu/~pal/pubs/mate.pdf>)
- [4] Xuan Long Nguyen, Tye Rattentbury: Localization algorithms for sensor networks using RF signal strength. (<http://www.cs.berkeley.edu/~xuanlong/cs252/sensor.ps>)
- [5] Kamin Whitehouse, David Culler: Calibration as Parameter Estimation in Sensor Networks (<http://www.tinyos.net/papers/p59-whitehouse.pdf>)
- [6] Kamin Whitehouse, Cory Sharp Eric Brewer David Culler : A Neighborhood Abstraction for Sensor Networks. (<http://compilers.cs.ucla.edu/emsoft05/WhitehouseSharpBrewerCuller04.pdf>)
- [7] Cameron Dean Whitehouse: The Design of Calamari: an Ad-hoc Localization System for Sensor Networks (<http://www.cs.berkeley.edu/~kamin/pubs/whitehouse02calamari.pdf>)
- [8] Chris Karlof, David Wagner: Secure routing in wireless sensor networks: attacks and countermeasures.(<http://www.cs.berkeley.edu/~ckarlof/papers/senroute-adnj.pdf>)
- [9] Kaan Yuksel, Jens-Peter Kaps, and Berk Sunar: Universal Hash Functions for Emerging Ultra-Low-Power Networks. (<http://www.crypto.wpi.edu/Publications/Documents/YukselKapsCnds04.pdf>)
- [10] David Gay, Philip Levis, David Culler, Eric Brewer: NesC 1.1 Language Reference Manual (<http://nesc.sourceforge.net/papers/nesc-ref.pdf>)
- [11] Philip Levis and Nelson Lee: TOSSIM - A Simulator for TinyOS Networks (<http://www.cs.berkeley.edu/~pal/pubs/nido.pdf>)
- [12] Philip Levis, Nelson Lee, Matt Welsh, and David Culler: TOSSIM -Accurate and Scalable Simulation of Entire TinyOS Applications (In Proc. Second ACM International Workshop on Wireless Sensor Networks and Applications (SenSys 03), 2003)
- [13] Deepak Ganesan: TinyOS and NesC (<http://www.cs.umass.edu/~dganesan/courses/fall04/slides/CS691AA-TinyOSNesC.pdf>)