Development of wireless sensor network for environment monitoring Applications

B. Karthikeyan
School of Electronics Engineering
VIT University, India
bkarthikeyan@vit.ac.in

R. Kumar
Wipro Technologies, India
Rajagopal.kumar@wipro.com

Srinivasa Rao Inabathini
School of Electronics Engineering
VIT University, India
israo@vit.ac.in

Abstract This paper presents a self-organizing wireless sensor network that can be used for environmental surveillance tasks, like wildlife behavioural studies. The parameters of interest that are measured and recorded are temperature, humidity, illumination and the motion of living objects inside the span of the network. An accurate time-synchronization algorithm has been implemented and tested successfully. A vital task during the whole development process was to keep the power consumption to a minimum, since the used nodes are battery powered. This paper shows that the TelosB mote combined with the TinyOS operating system provides an easy to implement infrastructure establishment. The TelosB has been extended with additional hardware, without disturbing the low power operation. The extension board can be fully controlled via software and has a sleep state where the current consumption drops to less than 0.1 A

Keywords: WSN; TelosB; TinyOS; Intrusion Detection; Time synchronization.

1 Introduction

The development of the human society in the last 100 years has significantly influenced the habitat of almost all mammals all over the world. As a result 22 percent of all known species were known to be globally threatened or extinct [1]. Habitat loss, affecting over 2,000 mammal species, was the greatest threat globally. A prime example was the population development of the Indian tiger. With an estimated population of 100000 in 1900, the value dropped by the factor 34 to less than 3000 in the year 2011 [2]. A key factor in preserving a species was to understand the behavior in their natural habitat. Wildlife behavioral studies were a crucial task to gather the necessary information. However, the observation by humans was restrained since the habitat might spread over a large area and the behavior of animals changed when humans were present. A method to overcome these problems was to deploy a network of electronic devices, each equipped with several sensors and a communication unit[17]. These so called wireless sensor networks (WSN) offer the opportunity to monitor large wildlife habitats without the intrusion and distortion by humans. The so collected data gave reliable information on the behav- ior of animals in their natural habitat. The sensors used were commercial of the shelf (COTS) to keep the cost of each node at a minimum. A gateway node that was connected to a device with Internet access gathers all collected sensor data and makes it available in real time to research groups all over the world. Furthermore a user interface that displayed the data in a user-friendly man- ner and offered the possibility to reconfigure the system has been designed. The work of this paper consisted of hardware- as well as software development. A vital task during the whole development process was to keep the power consumption to a minimum, since the used sensor nodes were battery powered.

2 Related Work

Wireless sensor networks are often mentioned as one of the key technologies of the 21st century. The high interdisciplinarity of the topic demands for improvements in various research areas. Low power electronics, energy aware routing algorithms, time synchronization and hardware miniaturization were just some of the disciplines that have to be mastered for the deployment of a functional WSN. Research prototype motes like the MicaZ [3] or the TelosB [4], both developed at the University of California, enable research groups to deploy and study real life sensor networks. Cerpa et al. [5] describe habitat monitoring as a driver application for wireless sensor networking. They proposed a tiered architecture for such applications and a Frisbee model[5] that optimizes energy efficiency when monitoring moving phenomenon. Mubarak et al. [6] showed a way for energy efficient intrusion detection, which could be transferred to the detection of a moving animal inside a habitat monitoring system. Their method could increase the network lifetime by using only few of the available sensors within a region of the WSN. A similar system, developed for border and perimeter security, was introduced by the Indian Institute of Science in 2010 and is called SmartDetect [7]. They use analog passive infrared sensors to detect human intrusions and reliably communicate such intrusions to the base station in a secure manner. The project focused on power optimal self-organization, reliable message delivery, security, low false alarm event detection, sleep-wake scheduling, energy efficient monitoring and debugging mechanisms. A prototype implementation of SmartDetect has been successfully field tested in an outdoor environment comprising of 25-30 nodes.Sanchez et al. developed a WSN for moving target monitoring in areas of special interest [8]. In particular, it has been applied for tracking animals approaching wildlife passages under roads. The system uses a combi-

nation of tracking capabilities, provided by infrared motion sensors, together with target identification through the use of camera sensors. They studied the effects of using different node layouts and densities with respect to system performance. Viani et al. have presented a WSN-based system for the monitoring of wildlife activity and the prevention of vehicle collisions in 2011 [9]. Their system consists of a network of sensors and actuators. Wildlife approaching the road gets detected with radar sensors and the drivers on the road get warned in real-time by means of light signal devices. The capabilities of the proposed system have been preliminarily assessed by means of an experimental setup. This system has been installed in a controlled environment located in Alps regions strongly affected by the problem of wildlife-vehicle collisions. A method for time synchronization in wireless sensor networks was proposed by Marti et al. in 2004 [10]. They analyzed all timing uncertainties that can occur during message delivery and came up with a protocol that they implemented successfully on the Mica [3] platform. Tests with real-world applications verified that the performance was markedly better than those of other existing time synchronization approaches on the same platform.

The achievements of all mentioned projects were used as a base for the development of a self-organizing WSN that uses COTS infrared sensors for the detection of intruder [16]

3 Methodologies

The first step was to analyze the major problems that will occur during the development process. The available low-cost infrared sensors could not directly be attached to the TelosB mote, due to their different supply voltages. A hardware that was able to match the supply voltages have to be developed. The major part of realizing this project consisted of software development. A first approach was to divide the whole project in a sum of subproblems. By doing this, it was possible to get a better overview over the numerous tasks that have to be fulfilled and the tools that can be used. The chart shown in Figure 1 was generated. The TelosB hardware was designed to run programs that were written in nesC for the operating system TinyOS. Besides TinyOS at least one more programming language has to be used. The system to the gateway node was connected must be able to receive the gathered information. This means it must run a dedicated software that listens to the serial port and dispatches the payload of the data packages.

In a real life sensor network the system to the gateway node was connected could be anything, from a headless Linux computer to an android smartphone with UMTS connection. Because of this, Java has been chosen as programming language, due to the portability provided by the Java Runtime Environment (JRE). This offers maximum flexibility for the hardware and the operating system that is used to inject the data into a network. The

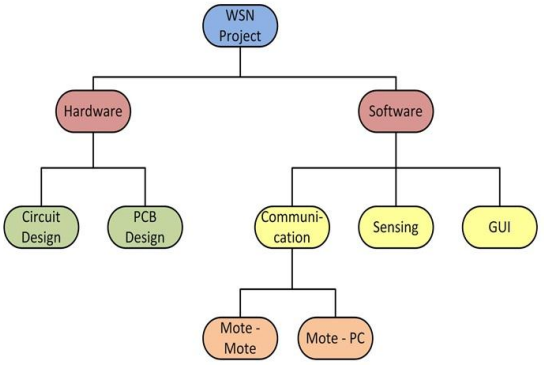

Figure 1: Division of the project into subproblems

sensor network has to fulfill numerous tasks, which are all interconnected to each other. The final system has to work reliable and must not fail because of its subsystems fails. The diagram in Figure 1 was further disassembled.

By this procedure it was possible to analyze the requirements for each subsystem specifically. Afterwards each of the subsystem passed through a trusted development process including its own phase of simulation and verification, like it is intended in the Waterfall model. The aim by using this sequential design process was to detect errors in early stages, so that they can be fixed by changing only the subsystem where the error occurred. If in a later process for example the communication and the sensing subsystem are interconnected, an error in the message delivery could also spread into the measurement process. The concentration on requirements and design before the code writing started also ensured minimal wastage of time. Linux has been chosen as environment for the network development, specifically Ubuntu 12.04 has been used. The tool chain consisted of Eclipse (Helios release), which was extended with a plug-in called Yeti 2 [11], so that it was possible to develop Java as well as TinyOS software with this SDK. The installed Java version was 1.6. TinyOS must be installed to work with the Yeti 2 plugin. For this work TinyOS version 2.1.1 was used, this includes the nesC compiler (ncc) version 1.2.4 and the msp430-gcc version 3.2.3. Also the Cooja [12] simulator was used to test the software in an early phase of development. With Cooja it is also possible to simulate huge sensor network deployments, consisting of hundreds of nodes.

4 Wireless Sensor Network Development

The parameters that should be measured and recorded by the sensor network are temperature, humidity, illumination and motion. The first three parameters could be measured with the TelosB directly. For motion detection the nodes were equipped with passive infrared sensors (PIR-sensor)[18]. All nodes gather information and after a given time interval, which is called collection period, they send a data packet to a gateway node. The region that

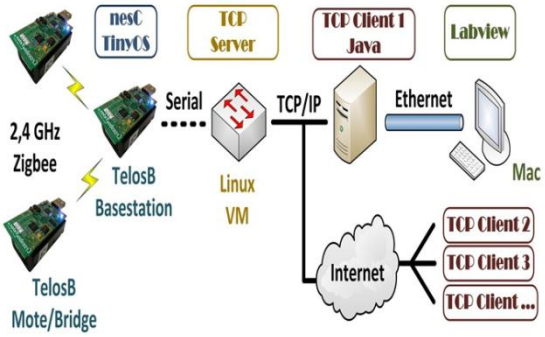

Figure 2: Overview of the whole measurement system

would be covered by the nodes are bigger than the radio transmission range of a single node. This meant that not all motes could reach the gateway node directly. To enable each node to send packets to the gateway, other nodes have to act as a bridge, meaning they relay incoming messages. The gateway node, which is also known as the root, forwards the data packets via serial interface to a computer with a storage device and a network connection. A program on the computer is listening to the serial port and receives the data of the root node. It makes all gathered information available through TCP/IP, so that clients can log into the server and also retrieve the data packets. As an example, a client Java application has been written. For each node that is sending data, a new file gets created and each incoming data packet gets appended into this file. The files allow the post data processing and analysis. Additionally a LabView program, which displays the data in a user-friendly manner, has been written. The program reads the data packets from the Java application and allows the examination of the parameters in realtime. Also with the LabView program one is able to send data into the network. So for example the collection period can be changed during the runtime of the program. Figure 2 shows a schematic of the whole system.

4.1 Hardware Development

The TelosB is powered with two AA batteries and therefore operates on 3 Volts. To operate the PIR-sensor module with the TelosB, a boost converter plug-in PCB was designed. It can get connected to the TelosB 10-pin expansion port, which also directly delivers the supply voltage. The module has a shutdown pin that can switch the boost converter on and off. The pin is connected to an output pin of the microcontroller. This allows for the development of software algorithms that switch on the motion detection sensor only when it is really needed, like in the Frisbee model [5]. The PIR-sensor output pin is connected to TelosBs MSP430 and can be used to trigger an interrupt on the MSP430 whenever a motion detected by the PIR sensor. A TelosB mote equipped with the boost converter board and a PIR-sensor can be seen in Figure 3.

Figure 3: TelosB mote equipped with boost converter and PIR sensor module

4.2 TinyOS

The motes must be able to measure the desired parameters and communicate with each other. As one of the nodes will be connected to a computer as a data sink, it will potentially have to fulfill more tasks than the other nodes[15]. The gateway node gets assigned with the ID 1. While the mote is booting it will check its number. The mote with number 1 configures itself automatically in a way that it can fulfill all the additional tasks of the base station. The network must be capable to communicate bidirectional. That means each node can send data to the gateway, and also that the gateway is able to send information to each node. This is useful to reconfigure the system and change the collection period. The collec- tion period is the time after which a node generates a data packet and initiates the message sending. The data collection of the nodes has been realized by the basic multi-hop networking abstraction of TinyOS, which is called tree collection. The motes organize themselve into a routing tree, centered on a particular mote, which is assigned as root. All messages that are generated flow automatically to the tree. The messages coming from the nodes that cannot reach the root directly get re- layed by the other nodes, which act as a bridge. The counter piece to the Collection protocol is the Dissem- ination protocol. It efficiently distributes a value across the network. The network got configured in a way that only the root node is able to do this. So the value of the collection period that the root node holds is the master value. By the use of Collection and Dissemination, the communication infrastructure in the network was estab- lished. Extensive tests with the Cooja simulator should verify the operability of the communication subsystem. The TelosB mote is equipped with two photodiodes that are connected to ADC ports of the microcontroller. The combined temperature and humidity sensor is connected via TWI, and the PIR sensor delivers its output signal to an I/O port, which got configured as digital input. To generate a data packet, all this three sources have to be

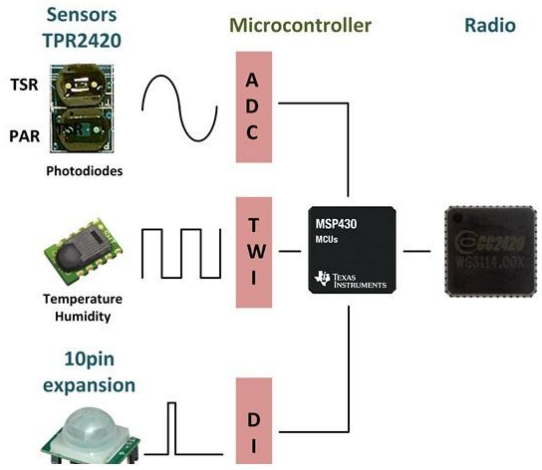

Figure 4: Data flow of the sensor reading

read out. A schematic of the information flow is shown in Figure 4. A message structure in the form of a TinyOS nx_struct has been defined to send all the collected sensor data to the gateway. The structure has variables to store temperature, humidity, two times illumination and the motion data. Besides these it also contains the ID of the sending node, which is necessary to assign the data packages later to its destination. Additionally it has one variable to store the time period over which these data got collectedThe diodes as well as the temperature and the humidity get read out with the TinyOS split phase interface Read. In case of the humidity the raw data is a 12-bit value, and in case of temperature it is a 14-bit value. The values have to be converted into an absolute value. Also sensor nonlinearities have to be compensated. In order to decrease the computational load on the battery powered nodes, this task was shifted to the gateway. That means, that the data packets with the sensor data that get send over the radio contain the raw measurement values. The gateway node sends this raw values over the serial port to the computer, where they finally get converted into absolute temperature and humidity values.

4.3 Low Power Consumption scheme

The values of the infrared sensor get read out with a pin that triggers an interrupt on the microcontroller. In the case of illumination, temperature and humidity, one value per collection period is sufficient, but in the case of the motion data, it is not accurate enough to receive only one value every 20 minutes. To achieve a higher resolution the motion sensor data gets encoded into a 32-bit variable. Since the output of the sensor is binary (either there is motion, or not), one bit is sufficient to store the current state of the sensor. The collection period gets divided into 32 parts, and so a resolution 32 times higher than the collection period can be achieved. Figure 5 explains the used scheme The scheme gets realized via a 32 bit mask which has the value 1 and gets shifted every

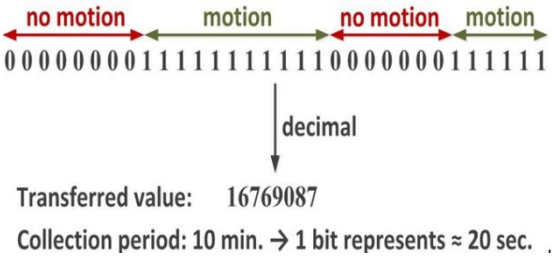

Figure 5: The values delivered by the motion sensor get coded into a 32 bit variable

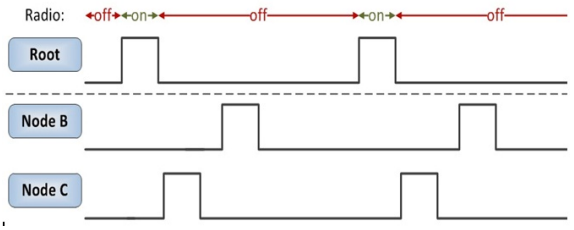

Figure 6: The communication windows dont match, so no communication is possible

(collection period / 32) seconds to the left by one. If the interrupt associated with the PIR-sensor gets triggered, a logical OR operation between the mask and the variable for the PIR values gets executed. As a result the 1 of the bit mask gets integrated into the PIR variable. While the radio service is switched on it will consume between 7.4 to 18.8 mA, additionally to the power consumed by the other components like microcontroller and sensors. This permanent power consumption is not acceptable for achieving long-term operation of the sensor network without changing the batteries. In order to extend the operational time of the system, the radio service is periodically switched on and off (duty cycled) to save energy. The collection time for the packets is somewhere in the range of 10-30 minutes. During this whole time the radio is turned off. After one collection period is over, the radio on each node gets turned on for a short time interval and all the generated packets get flooded to the root node. This synchronous low power message delivery scheme allows to maximize the energy savings [19]. The problem that occurs here was that during the deploy- ment of a sensor network each node gets switched on at a different time. As a result the timing windows of the on and off time of the radio are not overlapping. Figure
6 shows the problem.

4.4 Time Synchronization Algorithm

The synchronous switching of all nodes is only possible when there is a time synchronization between them. The timer of the root node gets defined as the global time scale, and all the other nodes have to synchronize with this scale. The synchronization has to be repeated periodically to compensate for clock drifting that will occur between the nodes. The reason can be slight differences in the resonance frequency of the quartz that is

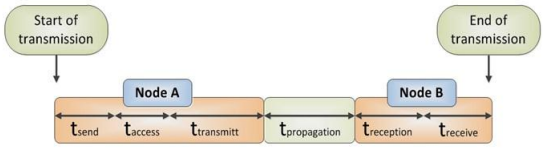

Figure 7: Timing delays that occur during the transmission of a radio message

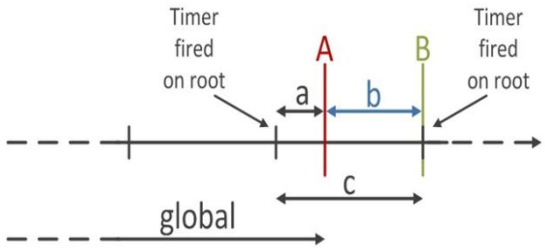

Figure 8: Detailed synchronization scheme clocking

the microcontroller. The drift effect gets even bigger when there is a temperature difference between the nodes, which in a large-scale sensor network could occur. Message sending over a radio channel contains various timing uncertainties. Some of them are strictly deterministic, others are highly non-deterministic. The delays have to be analyzed and compensated in order to achieve an accurate time synchronization between the nodes. Figure 7 shows a schematic of the most important delays that occur when a TelosB mote sends a message to another TelosB Mote. The meaning of various delay is discussed in [13]. The transmission time of the sender and the reception time of the receiver are overlapping To realize the time synchronization, a new message type TimeSyncMsg has been introduced. Each message consists of a single variable called globalTime. The User Button on the root node is wired to trigger an interrupt that initiates the time-sync process for the entire network. This process gets repeated automatically every hour. Once the process is initiated the global time is saved in the variable of the TimeSyncMsg instance. This time is used as the synchronization point. The message is then sent via the TimeSyncAMSend interface of TinyOS. It is similar to the AMSend interface, but has an additional parameter that can store the time of an event expressed in the local clock of the sending node. The receiver is able to read this time and translate it to its own local clock time via the TimeSyncPacket interface [14]. The interface has direct access to the MAC layer and performs a message time stamping during transmission and reception. This allows for the elimination of the highly non-deterministic delays. The only delay that is left theoretically is the propagation time, and as this is in the order of nanoseconds it can be neglected. Figure 8 shows a more detailed view of the synchronization scheme. The sync button on the root node gets pressed at point A. As mentioned above the TimeSyncAMSend interface allows to determine the time of point A expressed in the local time of the receiving node. This time will be called receiveEventTime in the following. The message contains also the global time of the root node, with this information one can calculate the clock offset between the sender and the receiver as shown in equation 1.

$$clockoffset = globalTime - receiveEventTime \quad (1)$$

This calculation is done immediately after the message has been received by the event handler. The clock offset is known for obtaining the absolute global time of the root node with equation 2.

$$globalTime = localTime + clockoffset \quad (2)$$

The next step is to estimate the time at which the timer on the root node will fire the next time, expressed in the local clock of the receiver. The length of c equals the collection period, with this information one can calculate the length of a with equation 3.

$$a = globalTime \ \% \ c \quad (3)$$

The remaining time period b until the timer fires equals c - a. One can now calculate the occurrence of event B with equation 4.

$$B = globalTime + c - a \quad (4)$$

Finally the time at which the timer on the root node will fire is known, expressed in the local time of the receiver. With this information a synchronous timer on the local node can be started. The synchronization process is completed with this step.

4.5 Serial Interface

The gateway node is sending all incoming data via serial interface to the computer. A possibility to acquire the data would simply be to listen to the serial port with a Java application. The problem that occurs when the application is directly listening to the serial port is that only one PC program can interact with the mote. Also it requires that the application, which is processing the data, is running on the PC that is physically connected to the basestation. The TinyOS SerialForwarder is a simple tool to get rid of these limitations. The program listens to the incoming packets at the serial port, and makes the data available over a TCP/IP stream. Applications that want to retrieve the packet data connect to the SerialForwarder, which acts as a proxy to read and write packets. This connection makes it also possible to connect a client over the Internet. The Message Interface Generator (MIG) is a tool that can generate a Java class that represents a nesC nx_struct, like the data packets that get send from each sensor node to the root. With this class and the net.tinyos.message package the

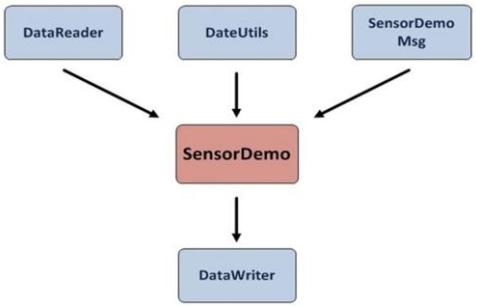

Figure 9: Structure of the Java program

data can able to send and receive TinyOS Active Message (AM) packets, whose payload is a value of the given nx_struct. The class also offers accessor methods to read the variables that are contained in the payload of the messages. The main task of the Java program is to process and store the incoming serial packets. The scheme of the Java program that has been written to solve all this tasks is shown in figure 9. Besides the MIG generated Java class for the handling of the AM packets, also classes for timestamping and file I/O are needed. Also it has to convert the raw data of temperature and humidity into absolute values and compensate the nonlinearities. The program generates a text file for each node from which it is receiving data packets. Every new incoming data packet gets timestamped and appended into the corresponding text file. Additionally the Java application can send serial messages to the gateway node. This message contains the value of the collection period. If the collection period gets changed the gateway node automatically initiates the dissemination process to spread the new changed value over the network. When the Java program gets started it needs a packet source as pa- rameter. With the Serial Forwarder running, this packet source is a TCP/IP port.

4.6 GUI

To display the data in a user-friendly manner and to offer real time data analysis, a LabView program with a graphical user interface has been written. The program reads the data from the Java program and displays it in a graph. The data of each node get displayed is shown in figure 10. Also the collection period can be changed out of the program. The software was a very useful tool during the development and can also be used in a real life application to get an overview about changing measurement values.

5 Results

5.1 Simulation

The communication system has been simulated and tested with the Cooja simulator. A simulation with 10

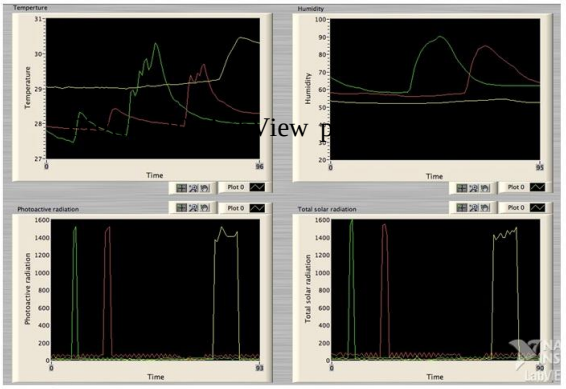

Figure 10: Lab View Program output

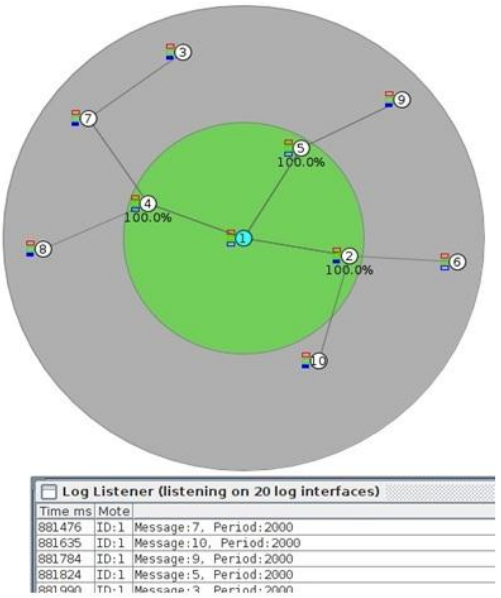

Figure 11: Traffic flow in a network consisting of 10 nodes

nodes proofed that multi-hop communication is working. The traffic flow of this simulation can be seen in figure 11

Only the nodes with the numbers 2, 4 and 5 can reach the root node (ID = 1) directly. The messages of all the other nodes have to be relayed by these nodes. As one can see in the log listener interface, the root node is receiving messages also from the nodes that are not in the direct range. The test messages contain the number of the sending node, as well as the collection interval, with that the nodes send the messages. The functionality has also been tested with a bigger network, consisting of 50 nodes.

5.2 Hardware

Tests with the developed PCB and the PIR-sensor showed that the current drawn during motion sensing is 220 A. If the sensor detects some motion and has to trigger the digital pin of the microcontroller, the value

Component	Mode	Current draw
Module	Active	1.8 mA
	Sleep	5.1 μA
RF Transceiver	Receive	19.7 mA
	Transmit (at 0 dBm)	17.4 mA
	Sleep	0.01 mA
Data from www.memsic.com and www.ti.com		

Figure 12: Power consumption of the TelosB mote

```
2012-11-28 20:55:28
ID 2
Temperature: 31,19 °C
Humidity: 55,29 %rH
Photoactive radiation: 34
Total solar radiation: 44
PIR Values: 00000011111111111111111111111000
PacketPeriod was 10 s
```

Figure 13: Linux console message generated by the Java program

increases to 1.2 mA. These are good values that fit to the rest of the power consumption of the TelosB mote, as shown in figure 12. The Java program generates a console output to inspect the incoming packets. Figure 13 shows the example of a console message. It contains all information that were included in the data packet, including the timestamp

6 Conclusion and future work

This paper presented the development of a wireless sensor network that can be used for wildlife monitoring, as well as other surveillance tasks. It showed that the TelosB mote combined with the TinyOS operating system provides an easy to implement infrastructure establishment. An accurate time-synchronization algorithm has been implemented and tested successfully. The TelosB could be extended with additional hardware, without disturbing the low power operation. The hardware can be seen as a universal extension board, which makes it easy to modify the function of the sensor network. It can be used for all sensors with operating voltages between 2,5 and 12 Volts, which have either a digital, analog, serial or TWI signal output. The board can be fully controlled via software and has a sleep state where the current consumption drops to less than 0.1 A. The gathered information gets collected from a data gateway and is available over the Internet to research groups all over the world. The system offers also a graphical user interface that displays the sensor data in a user-friendly manner, and offers the possibility to reconfigure the network.

As a future work the software could be extended with an algorithm to manage the infrared sensors. A possibility would be to deactivate all motion sensors that are completely enclosed by other motion detection ranges. In the moment, one of the neighboring motes detects a moving object, it sends a signal to activate all motion sensors that are close to it. The software could also be extended by an algorithm to determine the speed of a moving object that crosses through the network, by a simple distance per time calculation. Another way to improve the system would be to expand it with camera modules. Most of these modules need a 5 V supply, which could be delivered by the PCB. Since the camera consumes a lot of power compared to the PIR-sensor, it could stay switched of until the infrared sensor detects motion in the area. When that happens it switches the camera module on as long as there is some movement in the area. The combination of infrared motion detection and a camera offers a possibility for low power image capturing, which otherwise would not be possible.

7 References

[1] International Union for Conservation of Nature, An analysis of mammals on the 2008 IUCN Red List, Arizona State University, Texas AM University, University of Rome, University of Virginia, Zoological Society London. 2008, www.iucnredlist.org/mammals

[2] Wildlife Protection Society of India, The Indian Tiger - Quick Facts, S-25 Panchsheel Park New Delhi 110017, India, www.wpsi-india.org

[3] J. Hill and D. Culler, Mica: a wireless platform for deeply embedded networks, IEEE Micro, vol. 22, no. 6, pp. 1224, November/December 2002

[4] Joseph Polastre, Robert Szewczyk, and David Culler, Telos: Enabling ultra-low power wireless research, Computer Science Department, University of California, Berkeley, 2005

[5] Cerpa et al. Habitat monitoring: Application driver for wireless communications technology, UCLA Computer Science Technical Report 200023, December 2000

[6] Mubarak et al. Intrusion detection: An energy efficient approach in heterogeneous WSN, International Conference on Emerging Trends in Electrical and Computer Technology, 2011, IEEE 978-1-4244-7926-9

[7] The SmartDetect WSN Team, SmartDetect: An efficient WSN implementation for intrusion detection, Indian Institute of Science, Bangalore and Centre for AI and Robotics (CAIR) IEEE 978-1-4244-5489-1/10/

[8] Sanchez et al. Wireless sensor network deployment for monitoring wildlife passages, Sensors 2010, 10, 7236-7262; doi:10.3390/s100807236 ISSN 1424-8220

[9] Viani et al. WSN-based early alert system for preventing wildlife- vehicle collisions in Alps regions, ELEDIA Research Center @ DISI, University of Trento, 2011, IEEE 978-1-4577-0048-4/11

[10]Marti, Kusy, Simon, Ldeczi, The flooding time synchronization protocol, SenSys 04,November 3-5, 2004 ACM 1-58113-879-2/04/0011

[11] Benjamin Sigg, Yeti 2 - TinyOS plugin for Eclipse," http://tos-ide.ethz.ch/wiki/index.php

[12] Fredrik sterlind, A sensor network simulator for the Contiki OS, SICS Technical Report ISSN 1100-3154, November 2006

[13] Marti, Kusy, Simon, Ldeczi, The flooding time synchronization protocol, SenSys 04, November 3-5, 2004 ACM 1-58113-879-2/04/0011

[14] Marti, Sallai, Packet-level time synchronization, www.tinyos.net/tinyos-2.x/doc/html/tep133.html

[15] James Carnley, Bo Sun, S. Kami Makki "TORP: TinyOS Opportunistic Routing Protocol for Wireless Sensor Networks" 5th IEEE Workshop on Personalized Networks (PerNets 2011)

[16] Jochen Rust, Xinwei Wang, Rong Shen, Rainer Laur, Steffen Paul "Equidistant Piecewise Function Approximation for Neurocomputing Based Environmental Monitoring in Wireless Sensor Networks" Environmental Energy and Structural Monitoring Systems (EESMS), 2011 IEEE Workshop on 28-28 Sept. 2011

[17] Alin Argeseanu, Krisztina Leban, Ileana Torac "**matrix sensor for solar tracking systems**" Journal of Electrical engineering [www.jee.ro], Volume 9 / 2009 - Edition : 2

[18] Mohand saïd djouadi, razibaouene amir, zeghlache samir "Motion Detection and Tracking by Autonomous Mobile robot in Indoor Environment",Journal of Electrical Engineering [www.jee.ro] Volume 10 / 2010 - Edition : 3

[19] Pramod kumar, ashvini chaturvedi, shraddheya shrivastavaa "geographical location aware energy efficient routing scheme for query based wireless sensor networks " journal of electrical engineering [www.jee.ro] volume 13 / 2013 - edition : 2