An Adaptive Energy-conservation Scheme with Implementation Based on TelosW Platform for Wireless Sensor Networks

Liang Jin, Yi-hua Zhu
School of Computer Science and Technology
Zhejiang University of Technology
Hangzhou, Zhejiang 310023, China
yhzhu@zjut.edu.cn

Victor C. M. Leung
Department of Electrical and Computer Engineering
The University of British Columbia
Vancouver, BC, Canada V6T 1Z4
Email: vleung@ece.ubc.ca

Wen-Zhan Song
Department of Computer Science
Georgia State University
Atlanta, GA 30303

Email : songwz@cs.gsu.edu

Abstract— **Nodes in a wireless sensor network (WSN) are usually powered by batteries. Hence, it is important to efficiently expend the battery energy of each node in the WSN so that both runtime of the nodes and the lifetime of the WSN are prolonged. An event-driven energy-conserving scheme, called adaptive power-saving scheme (APS), is proposed. APS is able to adapt the sleep duration of a node to traffic variations. We have implemented APS based on TelosW motes, TinyOS, and NesC language. Experimental results show that APS outperforms the fixed time scheme (FTS) in terms of energy consumption and packet loss ratio.**

Keywords- Energy conservation; TelosW mote; Wake-On; Event-driven

I. INTRODUCTION

TinyOS is an open source and component-based operating system, which is implemented by NesC [1] (Network embedded system C programming language), a component-based programming language. It is designed for wireless sensor networks (WSNs) and supports various kinds of sensor motes such as TelosW, TelosB, Mica2, Intelmote2, and others. These kinds of motes operate with limited energy, small memory capacity, and low-speed processor to perform sensing and communications.

As a new member of the Telos mote family, TelosW enhances the functions of TelosB. The letter "W" in "TelosW" stands for wake-on [2], since the radio and some sensors embedded in a TelosW mote support a wake-on function. A TelosW mote is an ultra-low-power wireless sensor. Compared with a TelosB mote, a TelosW mote has the following new features [3].

1) A CC1101 radio chip is used, which can be woken up by radio signal without intervention of the mote's microprocessor, i.e., it supports wake on radio (WoR). In addition, it supports multiple data rates ranging from 1.2 Kbps to 500 Kbps while TelosB's radio only supports a fixed date rate of 250 Kbps.

2) It embeds an energy meter that can measure the input current and the input voltage so that the energy consumption of the mote can be determined in real-time.

3) The analog to digital converter (ADC) and light sensors in a TelosW mote also have wake-on capability. As many as 256 wake-on levels are allowed to be set for outside events to wake up the microprocessor.

In a WSN, nodes are mostly battery-powered. As a result, conservation of the nodes' battery power is extremely important for a WSN to run effectively. It should be noted that in some WSN applications, such as natural disaster monitoring, toxic gas monitoring, and more, the batteries of the nodes cannot be easily replaced or charged. Hence, how to efficiently expend the energy and prolong the life of nodes as well as that of the whole network becomes one of the most important and challenging issues, which is currently addressed by many researchers.

A node in a WSN is mainly comprised of four components: a sensor, a processing unit, a radio transceiver, and a power supply unit [4]. Energy consumption in a node is mainly from radio communications [5]. Therefore, it is critical to make the radio expend less energy. A TelosW mote makes use of drivers [6] to control radio communications. By controlling the active or sleep states of a node, various protocols including AlwaysOn, B-MAC [7], and X-MAC [5], can be realized. With AlwaysOn, the nodes are always active, i.e., they are not allowed to sleep even when they are idle. Obviously, it wastes energy.

Under the B-MAC protocol [7], a long preamble is received by all the neighboring nodes. After receiving the preamble, only the addressed node remains active to receive the packet while other nodes are allowed to sleep. As each node has to listen to the long preamble, the protocol has a disadvantage in energy consumption.

The X-MAC protocol [8] extends B-MAC and adopts several short preambles, each of which includes the address of a receiving node, to replace the long preamble in B-MAC. When a node receives a short preamble, it checks the address included in the preamble to see if it is the intended receiver. If not, the node goes to sleep after the short preambles; otherwise, it stays active and acknowledges to the sending node before the next short preamble so that the sending node

This work was supported by Natural Science Foundation of China under Grant No. 61070190, by Zhejiang Provincial Key Science & Technology Project of China under Grant No.2009C14033, and by Zhejiang Provincial Natural Science Foundation under Grant No Z1100455.

does not transmit the remaining short preambles. Usually, X-MAC expends less energy than the B-MAC.

In practice, two methods are usually used to assess the performance of a power saving technique: (i) use a simulation program to mimic the real situation, or (ii) by experimentation using physical devices. Method (ii) yields more accurate results, but faces the difficulty of instrumenting the nodes to measure their energy consumption. This is overcome in new devices such as the TelosW mote, which integrates an energy meter to measure energy consumption in real time. In this paper, we develop power-saving techniques applicable to the nodes in a WSN, and implement these schemes in TelosW motes for experimental evaluations.

The main contributions of the paper are: 1) proposal of an event-driven energy-saving scheme, called adaptive power-saving scheme (APS), which adapts to the traffic changes; and 2) experimental evaluation of an implementation of APS on TelosW motes.

The rest of the paper is organized as follows. Section II describes the crucial technologies of TelosW supporting power saving schemes. In Section III we introduce our proposed APS based on TelosW. Experimental results for APS are presented in Section IV. We conclude in Section V.

II. USAGE OF ENERGY METER IN TELOSW MOTE

The energy meter embedded in a TelosW mote adopts the technology of iCount [9], which is implemented by a sensor driver programmed in NesC. The value of iCount is obtained through the interface EnergyMeter provided by the component named EnergyMeterC. The following two statements in NesC can be used to read the value of iCount:

```
call EnergyMeter.init(); // initialize iCount counter
call EnergyMeter.read(); //get the value of iCount
```

It should be noted that, in the above code, the split-phase operation, defined in [10], is not needed for using the energy meter interface, although the interfaces of other sensors (e.g., light sensor) usually do.

Using the value of iCount, the total energy consumption, denoted by E_{cost}, can be computed as follows [9]:

$$E_{\cos t} = \frac{1}{a} 10^{(\log_{10} iCount - b)} = \frac{iCount}{a 10^{b}} \quad (1)$$

where a and b are two parameters whose values are related to the battery voltage. For instance, a=1.06 and b=6.07, if the battery voltage is between 3.05V and 2.95V [9]. It can be clearly seen from (1) that E_{cost} has a linear relationship with iCount.

III. POWER-SAVING SCHEMES BASED ON TELOSW

For WSN applications in which sensor nodes are used to capture periodic events, to save power, a sensor node alternatively switches between the active state and the sleep state on a fixed time intervals basis. That is, the following process is performed repeatedly: it stays in the active state for a preset period and then switches to and stays in the sleep state for another preset period. We refer to this scheme as the *fixed time scheme* (FTS). Figure 1 depicts a timeline of FTS, in which a rectangle represents the time periods in which the power-saving node stays active, i.e., the node wakes up at time A and stays awake between times A and B, sleeps between times B and C, wakes up at time C again, and so on.

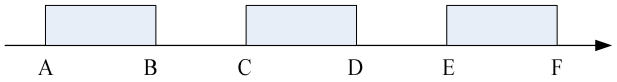

Figure 1. Timeline of the FTS

In fact, FTS exhibits an obvious weakness. If the time interval is too small, the nodes are very busy in changing their states, causing much more extra energy to be expended, and besides, the correlation of two successive data packets is too high so that lots of the sensed data are useless. If the time interval is too large, the changes of the monitored object cannot be captured in time and it is with high probability that some important events are missed. Therefore, an ideal power saving scheme should adapt to the monitored objects.

Since a TelosW mote embeds an energy meter and can be set to 256 different wake-on levels to wake up its microprocessor or radio, it is suitable for energy efficient event-driven applications. Therefore we develop an *adaptive power-saving scheme* (APS) based on the TelosW mote, in which traffic is taken into account so that the sleep duration of the power saving node adapts to traffic variations. In the proposed APS, we use the exponential moving average to predict the traffic arrival intervals (TAIs):

$$T_{n+1} = \theta T_n + (1-\theta) A_n, n = 1,2,\cdots \quad (2)$$

where T_n denotes the n-th predicted TAI, A_n stands for the n-th real TAI, and θ is a smoothing factor taking values in [0, 1]. Initially, T_1=A_1. The smoothing factor θ impacts the predicted TAI as follows: it emphasizes the recent traffic variations if θ is set to a value close to 0, whereas it reflects the past traffic or historic traffic if θ is set close to 1. Hence, in the case that traffic is changing, θ close to 0 is preferred so that APS can be geared to the variations of the traffic.

IV. EXPERIMENTS WITH POWER-SAVING SCHEMES

A. Power-saving schemes used in a parking lot

First, we introduce the TelosW motes, which are used in the experiment, in a little more details. The layout of a TelosW mote is depicted in Figure 2 while the picture of a real mote is shown in Figure 3.

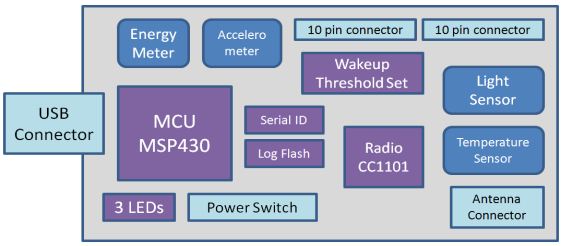

Figure 2. The layout of a TelosW mote [6]

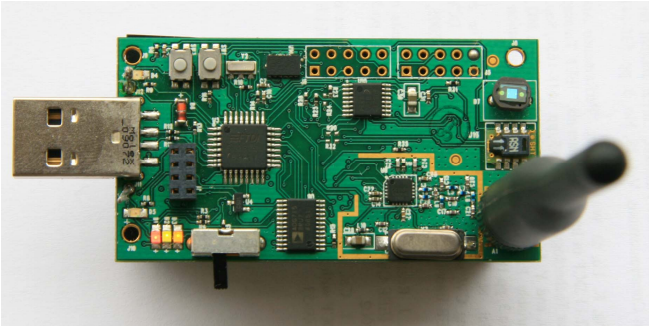

Figure 3. A real TelosW mote

Next, we present an experiment based on TelosW motes to test our power saving schemes, in which a simple WSN is used to monitor a basement parking lot that has an entrance and a nearby exit.

As shown in Figure 4 (a), we place a TelosW mote at one side of the entrance of the parking lot, and a light source is fixed on the other side, i.e., point A in the figure (the mote deployment for the exit of the parking lot is omitted here as it is similar to that for the entrance). In the sequel, we will refer the mote placed at the entrance as the *entrance mote* and the one at the exit as the *exit mote*. Either the entrance mote or the exit mote is called a *gate mote*. The topology for the WSN, consisting of TelosW motes, is shown in Figure 5, in which the relay nodes are used to forward packets to the sink node, which forwards them to the personal computer (PC).

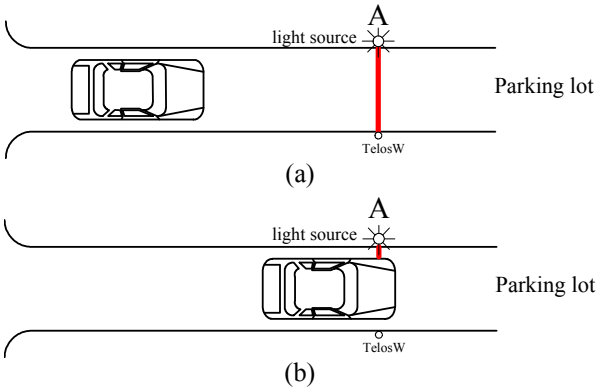

Figure 4. A TelosW mote fixed in the entrances of the parking lot

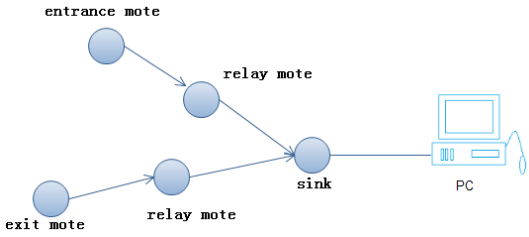

Figure 5. The topology of the WSN

We set a threshold for the light sensors of the motes so that the microprocessor stays asleep if the light intensity captured by a sensor is larger than the threshold whereas it wakes up as soon as the light intensity is less than the threshold. Needless to say, the light source is required to emit light, which points to the light sensor, with intensity larger than the prefixed threshold. Thus, if no car drives into or out of the parking lot, the light sensor of the mote is lit by the light source, keeping the microprocessor and the radio of the mote asleep. However, when a car drives into or out of the parking lot, it will block the light (cf. Figure 4(b)) briefly, causing the light intensity captured by the sensor to be lower than the threshold, which triggers the gate mote to generate a hardware interrupt to wake up the microprocessor. Then, the event related to the car moving into or out of the parking lot is reported to the PC via the relay motes and the sink. Moreover, the temperature and humidity in the parking lot, together with the battery voltage of the gate motes, can also be delivered as needed to the relay node, then to the sink node and to the PC. Obviously, the number of the cars kept in the parking lot can be derived from the number of the cars captured by the entrance mote minus that captured by the exit mote.

To conserve energy, the relay motes and the sink are in sleep state initially. The radio signal from the entrance or exit mote wakes up the relay motes, which in turn wakes up the sink by radio signal. Clearly, the relay motes and the sink have less chance to sleep to save power, if the gate motes transmit to the relay motes frequently. Hence, to conserve the energy of the relay motes and the sink, as well as the gate motes, the gate motes should not transmit too frequently. Noting that the information about the parking lot needs not be provided in real time, i.e., it can tolerate some delay, we apply FTS and APS power saving schemes, in which FTS and APS are run in the gate motes to control when the gate motes transmit packet to the relay motes so that the relay motes, as well as the sink mote, can sleep longer to save power.

Assume Γ is the delay limitation that allows a gate mote to defer to transmit a packet to the relay mote. In other words, the gate mote can keep a packet for the period no more than Γ. Each of the gate motes allocates a buffer to hold the data sensed during the sleep duration of the respective relay mote. Now, we describe how to use FTS and APS:

a) Under FTS, the relay motes are allowed to sleep, with both the sleep and the awake durations of the relay motes set to Γ. Besides, the gate motes transmit to the relay motes every Γ time. Moreover, the sink is woke up by the radio signal from the relay motes and sleeps as soon as it receives all the packets from the relay motes.

b) Under APS, a gate mote makes use of (2) to predict the arrival interval of cars so that it can buffer packets as many as possible within delay limitation Γ. The gate mote transmits its buffered packets to the relay mote, which is woke up by the radio signal from the gate mote, if it predicts that the next TAI plus elapsed time will exceeds delay limitation Γ. Figure 6 illustrates the proposed APS.

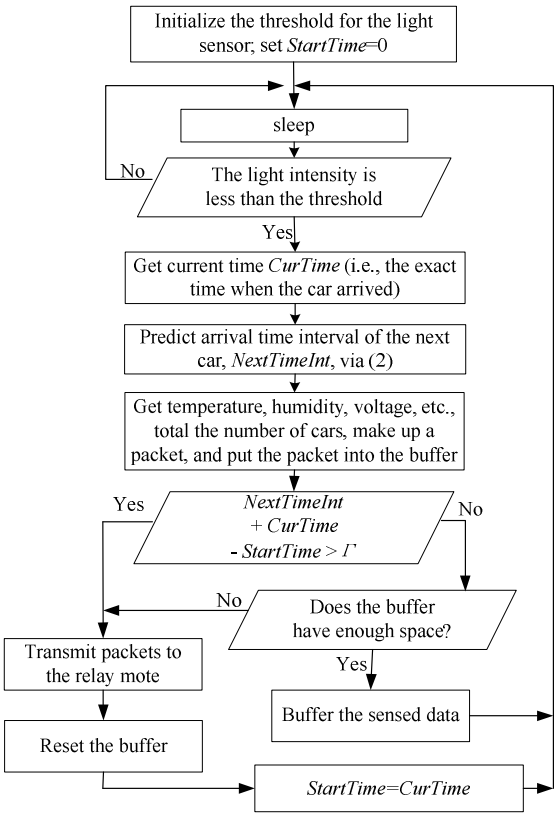

Figure 6. The APS for the gate mote

B. *Implementation based on TinyOS and nesC*

We use nesC in TinyOS 2.0 to realize the power saving schemes described in the previous section. The message packet structure is presented in Table I, in which the detailed content of the field "payload" is defined in Table II, where the field "number of cars" is incremented by the number of cars entering the parking lot as reported by the entrance mote and decremented by the number of cars leaving the parking lot as reported by the exit mote.

TABLE I. MESSAGE

Offset (Byte)	Field
0	type
1-2	address of the destination
3-4	address of the link source
5	length of packet
6	group ID
7	type of the active message handler
8-21	payload

TABLE II. CONTENT OF THE PAYLOAD IN MESSAGE

Offset in the payload field (Byte)	Field
0	node ID
1	number of cars
2-5	value of iCount
6-7	temperature
8-9	humidity
10-11	battery voltage
12-13	TAI

We use the following structure in nesC to represent the payload field in the above message packet:

```
typedef nx_struct {
    nx_uint8_t  id;              //node ID；
    nx_uint8_t  NumCar;          //number of cars
    nx_uint32_t  energy;         //iCount；
    nx_uint16_t  temperature; // temperature；
    nx_uint16_t  humidity;  // humidity;
    nx_uint16_t  voltage;     // battery voltage；
    nx_uint16_t TAI;          // interval arrival time of cars;
} Msg;
```

In order to observe the working status of each TelosW mote in the WSN, the three LEDs mounted on the mote are used (cf. the left-bottom corner of Figures. 3-4). We name them from the left to the right as LED0, LED1, and LED2, respectively. When they are on, LED0, LED1, and LED2 emit light in red, green, and orange, respectively. If a car drives into the parking lot, the entrance mote generates a hardware interrupt, which is observed by toggling LED2. If a packet is successfully transmitted to a neighboring mote, LED0 will be toggled; if a mote receives a packet, LED1 will be toggled.

The application integrated with the power-saving schemes is composed of a couple of nesC components. Apart from the useful components provided by TinyOS [10] such as MainC, LedsC, ActiveMessageC, AMReceiverC, AMSenderC, and SensorC, which are wired to form the application, more components with detailed implementations are needed. Here, we only present some key components and implementations.

To implement the FTS and the APS with delay limitation Γ, we need a buffer to hold the sensed data so that the relay motes as well as the sink can sleep longer to conserve energy. Hence, we define the following interface.

```
interface PowerSavingBuffer {
    // check if the buffer is full
    command  bool  IsBufferFull() ;
```

```
    // adds a packet to the buffer
    command  void  PutTobuffer(message_t ptr) ;

    // resets the buffer
    command void ResetBuffer();

    //to get the number of packets in buffer
    command uint8_t BufferNum();
  }
```

Then, we develop the component PowerSavingBufferP, in which the structure of the buffer is defined as the same as that of a packet. Additionally, for the gate motes, which are woke up by light sensor, we present the following module.

```
module PowerMeterP {
  uses{
       interface PowerSavingBuffer;
       interface …    // some interfaces are omitted here.
   }
}
implementation {
   event void Boot.booted() {
       //set the threshold for the light sensor
       call WakeupLight.setThreshold(1, 0x90);
  }
   //set events
   async event void WakeupLight.adc_int(){
       The logic shown in Figure 6 is implemented here.
     In addition, we use the following functions to read
     temperature, humidity, energy of battery:
     TemperatureRead.read(), HumidityRead.read(),
     and BatteryRead.read(), and EnergyMeter.read().
   } //end of the implementation
```

V. PERFORMANCE OF THE POWER SAVING SCHEMES

In this section, we conduct some experiments to observe the performance of the proposed power-saving schemes, i.e., FTS and APS. Considering that we need too much time to obtain the outcomes of the experiments if we use real cars to trigger the light sensors of the gate motes, and that the cars entering and leaving the parking lot usually obey a Poisson process, i.e., the arrival intervals of cars forms a stochastic process that obeys a exponential distribution, in order to get the outcomes in a shorter time, we, generate TAIs of the gate motes according to an exponential distribution with rate λ. Noting that the sink node connects to the PC directly and the sink node can be powered by the computer, we do not take the energy consumption of the sink into account in the experiment. Besides, Γ and θ are set to 600 s and 0.3, respectively.

We investigate the case that the average interval between two successive car arrivals is 2 minutes. That is, the rate parameter λ of the exponential distribution which generates TAIs is set to 1/120 car per second. The experiment with duration of four hours leads to Figure 7, which compares APS with FTS for the total energy cost, i.e., total iCount. This figure, which depicts iCount values when the experiment time reaches 30, 60, 90, 120, 150, 180, 210, and 240 (minute), indicates that: 1) APS outperforms FTS in terms of energy cost; and 2) the difference of the energy expended by APS and FTS grows with the increase of time. In other words, the longer APS used, the more energy is saved.

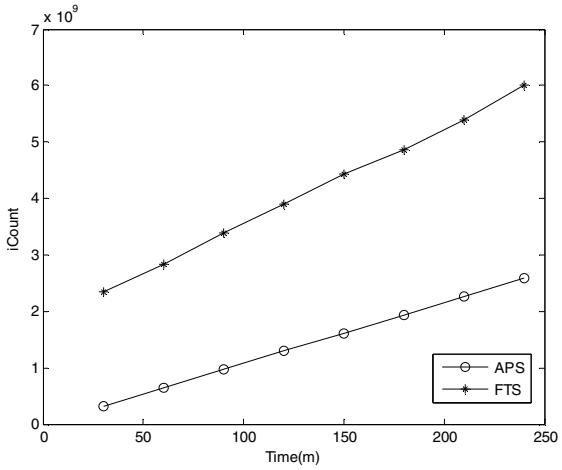

Figure 7. Energy consumption comparison of APS and FTS

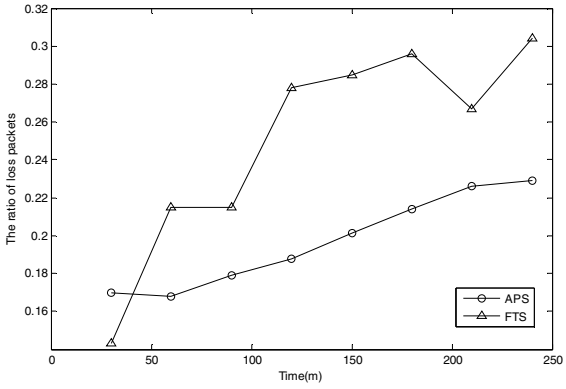

Figure 8. The loss of packets comparison

As well known, power saving schemes that let motes sleep usually introduce packet losses, especially when a longer sleeping period and a small buffer size are used. To investigate packet loss, we use two motes, a gate mote and a relay mote, to do the experiment. The comparison of the packet loss rate between APS and FTS is shown in Figure 8, in which the experiment parameters are the same as that for Figure 7 except that the size of the buffer is set to 10 packets and the gate mote retransmits at most 3 times if it does not receive the ACK packet from the relay mote. Figure 8 shows that: 1) APS exhibits a lower packet loss rate than FTS as time progresses; and 2) the packet loss rate of FTS fluctuates with time, whereas the APS changes smoothly and

approaches a steady level gradually. The reason is that, as shown in Figure 7, under APS, when a packet arrives, each gate mote checks its buffer to see if there is any available space to buffer the packet, and the gate mote immediately transmits its buffered packets to the corresponding relay mote if it finds no room in the buffer for the arriving packet. Noticeably, there is a cross-point in Figure 8, which reveals that the ratio of loss packets under APS is larger than that under FTS at first (cf. the result in Figure 8 when the experiment time is set to 30 minutes), but this situation is quickly changed as the experiment progresses (cf. the result in Figure 8 when the experiment time reaches 60 minutes or more). In other words, the cross point indicates that APS is able to adapt to the variation of TAIs so that the ratio of loss packets is reduced.

VI. CONCLUSION

In this paper, based on the new features of TelosW motes, i.e., embedded energy meter and wake-on capability, we have proposed a novel power saving scheme, the adaptive power-saving scheme, for WSNs. APS is driven by events and is able to prolong the runtime of motes as well as the life of the WSN. We have provided an experimental evaluation to show that this scheme can be applied in practice. In future, we will be further investigating the impacts of parameters Γ and θ on the performance of the proposed APS. Moreover, we will consider embedding learning strategies such as reinforcement learning into APS so that it can achieve better performance in power saving while it well adapts traffic changes.

REFERENCES

[1] D. Gay, P. Levis and D. Culler, "Software design patterns for TinyOS," In Proceedings of the 2005 ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES), P 40–49, New York, NY, USA, 2005. ACM Press.

[2] G. Lu, D. De, W.-Z. Song, and B. Shirazi, "A Wake-On Sensor Network," SenSys'09. November 4–6, 2009, Berkeley, CA, USA.ACM 978-1-60558-748-6.

[3] G. Lu, D. De, M. Xu, J. Cao, W.-Z. Song, "TelosW: Enabling Ultra-Low Power Wake-On Sensor Network", The Seventh IEEE International Conference on Networked Sensing Systems (IEEE INSS 2010)

[4] V. Raghunathan, C. Schurgers, and P. Sung, "Energy-aware wireless microsensor networks," IEEE Signal Processing Magazine, vol.1, no.7, pp. 40-50, Mar. 2002.

[5] S. D. Murugantha, D.C.F. Ma, and A.O. Fapojuwo, "A centralized energy-efficient routing protocol for wireless sensor networks," IEEE Communications Magazine, vol.43, no.3, pp. s8-13, Mar. 2005.

[6] "TelosW: an ultra-low-power wireless mote with wake-on", http://sensorweb.cs.gsu.edu/research/TelosW.html, last visited:10-03-2010.

[7] J.Polastre, J. Hill and D. Culler, "Versatile low power media access for wireless sensor networks," Proceedings of the 2nd international conference on Embedded networked sensor systems, November 03-05, 2004, Baltimore, MD, USA.

[8] M. Buettner, G. Yee, E. Anderson and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," Proceedings of the 4th international conference on Embedded networked sensor systems, October 31-November 03, 2006, Boulder, Colorado, USA.

[9] P. Dutta, M. Feldmeier, J. Paradiso and D. Culler, "Energy Metering for Free: Augmenting Switching Regulators for Real-Time Monitoring," In Proceedings of 7th International Conference on Information Processing in Sensor Networks (IPSN'08), 2008.

[10] P. Levis, "TinyOS Programing", http://csl.stanford.edu/~pal/pubs/tinyos-programming.pdf, last visited:10-03-2010.