MobiSense: Mobile Body Sensor Network for Ambulatory Monitoring

AGUSTINUS BORGY WALUYO, WEE-SOON YEOH, ISAAC PEK, YIHAN YONG, and XIANG CHEN
Institute for Infocomm Research (I^2R), A*STAR, Singapore

This article introduces MobiSense, a novel mobile health monitoring system for ambulatory patients. MobiSense resides in a mobile device, communicates with a set of body sensor devices attached to the wearer, and processes data from these sensors. MobiSense is able to detect body postures such as lying, sitting, and standing, and walking speed, by utilizing our rule-based heuristic activity classification scheme based on the extended Kalman (EK) Filtering algorithm. Furthermore, the proposed system is capable of controlling each of the sensor devices, and performing resource reconfiguration and management schemes (sensor sleep/wake-up mode). The architecture of MobiSense is highlighted and discussed in depth. The system has been implemented, and its prototype is showcased. We have also carried out rigorous performance measurements of the system including real-time and query latency as well as the power consumption of the sensor nodes. The accuracy of our activity classifier scheme has been evaluated by involving several human subjects, and we found promising results.

Categories and Subject Descriptors: J.3 [**Life and Medical Sciences**]: *Health*

General Terms: Design, Performance

Additional Key Words and Phrases: Wireless health system, ambulatory patient monitoring, pervasive healthcare, wireless body sensor network

ACM Reference Format:

Waluyo, A. B., Yeoh, W.-S., Pek, I., Yong, Y., and Chen, X. 2010. *MobiSense*: Mobile body sensor network for ambulatory monitoring. ACM Trans. Embedd. Comput. Syst., 10, 1, Article 13 (August 2010), 30 pages. DOI = 10.1145/1814539.1814552 http://doi.acm.org/10.1145/1814539.1814552

A. B. Waluyo is currently affiliated with Monash Unitversity, Australia.

The work for this article was supported by the Science and Engineering Research Council (SERC) of the Singapore Agency for Science, Technology, and Research (A*Star) on Embedded and Hybrid Systems II (EHS-II) under grants 052-118-0058 and 052-118-0052.

Authors' addresses: A. B. Walue (corresponding author): agustinus.borgy.waluyo@manash.edu; W-S. Yeoh, I. Pek, and X. Chen: {wsyeoh, ipek, xchen}@i2r.a-star.edu.sg; Y. Yong: y.yihan@gmail.com.

DOI 10.1145/1814539.1814552 http://doi.acm.org/10.1145/1814539.1814552

1. INTRODUCTION

The proliferation of wireless technology and recent advances in miniature sensors have facilitated remote patients' treatment and ambulatory health monitoring that brings a significant impact to the healthcare industry. This phenomenon is just in time because of the imminent crisis in healthcare systems due to reasons such as economic, social, and demographic trends. According to the U.S. Bureau of the Census, the number of elderly over age 65 is expected to double from 35 million to nearly 70 million by 2025 when the youngest baby boomers retire [U.S. Census Bureau 2009]. This trend is global, and hence the worldwide population with average age 65 is expected to expand more than double from 357 million in 1990 to 761 million in 2025. In conjunction with this, the number of heart-related diseases may increase accordingly. This, highlights the need for better health care solutions.

Most governments world-wide have stated healthcare as one of the top priority agendas to address [Yang et al. 2004]. One way to improve healthcare services is through the deployment of a wireless body area network (WBAN). A WBAN typically consists of a gateway and number of sensor devices strategically attached to the human body [Lo et al. 2005; Lo and Yang 2005].

These sensors are capable of sampling, processing, and communicating one or more vital signs such as heart rate, blood pressure, oxygen, saturation, and activity, or environmental parameters including temperature, humidity, and light (Estrin et al. 1999). WBAN opens up many exciting opportunities for medical caregivers to monitor patients' conditions in real time, anytime and anywhere.

A recent report stated that one-quarter of the population in the developed countries suffer from cardiovascular disease (CVD) [Jamison 2006]. Electrocardiograms (ECG), which capture brief periods of clinical monitoring in the hospital are not able to sample a patient's critical rare slight events, measure physiologic responses during a patient's normal daily activities, or capture the circadian variation in physiologic signals. Thus, ambulatory monitoring of patients is of much importance in the healthcare domain.

In light of this, we present *MobiSense*, a novel mobile healthcare system for monitoring ambulatory patients in a WBAN environment. The proposed system is designed to continuously monitor patients' activities and the corresponding heart status, for an extended period of time, and subsequently relay critical information to emergency services for a timely medical response. Furthermore, our system is built with important features such as critical-self-wake, reconfiguration, as well as resource control and management schemes. As both the sensor and host device in our system reside in a resource-constrained environment, it creates significant challenges. These challenges include *resource management and optimization*, *optimized system design and coding*, while consistently providing *high-level of accuracy*. Each of these issues is addressed in this article.

This system is part of a larger project called Embedded and Hybrid System II (EHS-II) funded by the Agency of Science, Technology, Research (A*STAR), Singapore [Embedded and Hybrid Systems II 2009]. EHS-II is comprised of

various interrelated projects, each of which reflects a component in the body sensor network technology ranging from the base-band, RF, and MAC layers to various applications. This article is an extension of our preliminary work that has been reported in Waluyo et al. [2008] and Yeoh et al. [2008]. Following this initial work, we have successfully carried out system integration, test-bedding and prototype development in which details are presented in this article. The main contributions of this article are threefold.

(1) A holistic framework of the proposed mobile health monitoring system, MobiSense, is presented. It starts with the overall architecture of the system from the sensor to the application layer, which includes the components, functionalities, and communication among components; followed by the algorithms and processes to derive the user's activity and heart rate.
(2) The proposed framework with all the features has been developed and implemented in a real-world environment. The prototype of the system is shown and described in detail.
(3) System evaluations based on the developed prototype have been performed. The measurements include power consumption of the sensor nodes, real time performance, and query latency. The effective use of our critical-self-wake scheme and activity classifier have also been investigated.

The remaining sections of this article are organized as follows. Section 2 presents the existing work on health monitoring systems while incorporating sensor devices. Section 3 discusses MobiSense's system design and architecture, including its functionalities, features, and algorithms. Following this, the prototype of the system is presented in Section 4 along with some system evaluations based on the prototype that has been developed. Finally, Section 5 concludes the article. It should be noted that the terms user, wearer, and patient in this article are used interchangeably.

2. PRIOR ARTS

The emergence of miniaturized sensors, embedded microcontrollers, and radio interfaces on a single chip to name a few, have introduced a new generation of wireless sensor networks suitable for various applications. Such applications include habitat monitoring [Szewczyk et al. 2004], traffic patterns and navigation, plant monitoring in agriculture [Burrell et al. 2004], and healthcare [Jovanov et al. 2003] as one of the most exciting application domains. Focusing on healthcare, wearable sensor devices, and the associated applications can be further classified into five categories namely: health monitoring, recovery, emergency services, homecare, and rehabilitation as depicted in Figure 1.

Starting with wearable sensor devices, it is worth noting that there has been a significant increase in the number of such devices, ranging from simple pulse monitors [Polar 2009], activity monitors [Actigraph 2009], portable Holter monitors [Med-Electronics 2009], to sophisticated implantable sensors [Cardio Micro Sensor 2009]. Holter monitors, as an example, have been used to simply collect data in traditional medical monitoring systems. In this case,

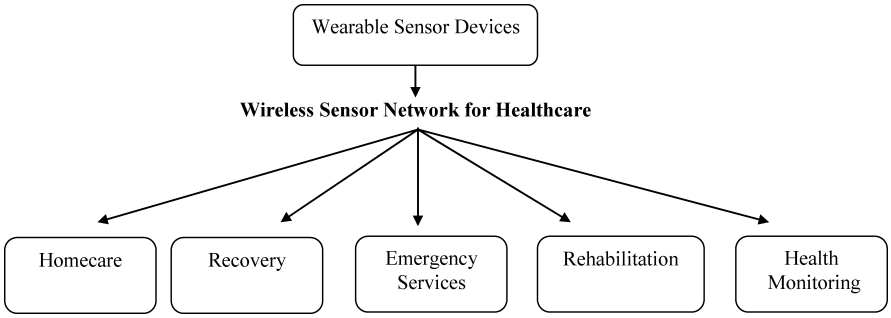


Fig. 1. Wireless sensor network for healthcare.

the data processing and analysis are performed offline; thus the deployment of such devices is impractical for continued monitoring and early detection of medical disorders. Most systems with multiple sensors for physical rehabilitation typically employ wires to connect the sensors with the monitoring system. These wires may limit the patient's activity and level of comfort and thus may have negative feedback [Martin et al. 2000]. Wireless sensor networks are therefore seen to be the solutions of this issue. The following discusses the deployment of wireless sensor networks in various healthcare applications.

Ross [2004] has illustrated homecare applications as part of the smart home system. The home is installed with a network of wirelessly linked sensors attached to nearly all of the home appliances. They are looking into probing the network remotely to monitor patients with dementia and other ills of aging and use the information to help the patients' families care for them. The next generation of older people may live in a world whereby every beat of their hearts and their daily activities are monitored, analyzed, and evaluated for signs of abnormalities. Such care may actually be less intrusive than forcing people to leave their own homes for nursing homes, which causes the loss of independence. The European E-care system is designed to monitor patients with short-term (1–2 months) recovery from treatment as well as those with chronic or long term illnesses such as diabetes. This system is also useful for the elderly people who live a normal life but at the same time require constant attention to the state of their health [Marsh 2002]. In Australia, a company named ProMedicus has designed a secure messaging system for transmitting pathology results to over 19,000 doctors and it has become a success story in IT Healthcare [ProMedicus 2009].

In another application, some wearable wireless sensor devices have been deployed for emergency services such as fall detection [Wang et al. 2008]. This system utilizes a triaxial accelerometer and is able to detect eight different kinds of falling postures. If falling is detected, an alarm will be activated. Other systems have been developed for computer-supervised rehabilitation for various conditions including early detection of medical conditions, for example heart monitors that are able to warn users about impending medical conditions [Welch et al. 2004] and provide information to a specialized center in the case of catastrophic events [Malan et al. 2004]. In a similar application,

Table I. Comparison of Health Monitoring Systems

Health monitoring system	WBAN	Activity classification (postural and movement) using accelerometer sensors	ECG waveform rendering and heart rate detection	Mobile platform	Resource management and reconfiguration
MIThril	–	√	√	√	–
CodeBlue	√	–	√	√	–
CardioNet	–	–	√	√	–
NASA	–	–	√	√	–
Mobisense	√	√	√	√	√

accelerometer-based monitoring of physical activity with feedback that can improve the process of physical rehabilitation has also been reported in Mathie and Celler [2001], and Zhou and Hu [2004].

Mobisense, our proposed system, is a health-monitoring application especially for activity, ECG, and heart-rate monitoring of ambulatory patients for further analysis by caregivers. The system resides in a resource-constrained mobile device and communicates with multiple wireless body sensor devices. A number of research efforts on health monitoring have been reported, for example, MIThril and CodeBlue. MIThril was initiated by researchers at the MIT Media Lab [DeVaul et al. 2003]. They have developed a wearable computing platform for ECG, skin temperature and galvanic skin response (GSR) sensors that is compatible with both custom and off-the-shelf sensors. MIThril is mostly deployed to investigate human behavior recognition as well as to create context-aware computing interfaces [Pentland 2004]. CodeBlue, developed by Harvard University, focuses on developing wireless pulse oximeter sensors, wireless ECG sensors, and triaxial accelerometer motion sensors. These sensors can be used on patients in the hospital to transmit vital signs to caregivers via an ad hoc network [Lorincz et al. 2004]. Others like CardioNet offer a remote heart monitoring system, where ECG signals are transmitted to a PDA (personal digital assistant) and then routed to the central server by using the cellular network [Ross 2004]. A real-time remote arrhythmia monitoring system prototype developed at NASA has been designed to collect real-time electrocardiogram signals from a mobile or homebound patient, combines them with GPS location data, and transmits this information to a remote station for display and monitoring [Liszka et al. 2004].

Mobisense is distinguished from these existing health monitoring systems as highlighted in Table I.

3. MOBISENSE: PROPOSED MOBILE BODY SENSOR NETWORK

This section discusses the architecture, components and functionalities of the proposed MobiSense system. We are starting with an overview of the ambulatory monitoring application. The ambulatory health monitoring as depicted in Figure 2 is comprised of four components, namely the wearer/patient, body sensor devices, a central/Web server and doctor/caregiver application in a mobile device. MobiSense, as a client-side application, is concerned with the first two components only.

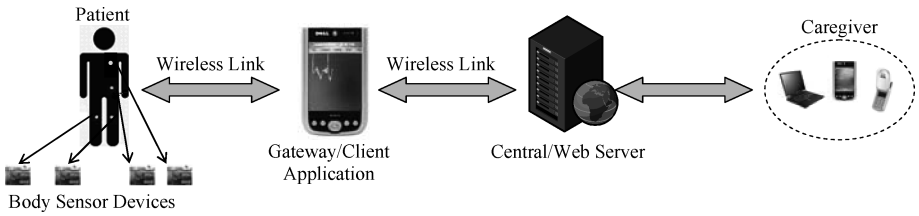


Fig. 2. System overview of ambulatory health monitoring.

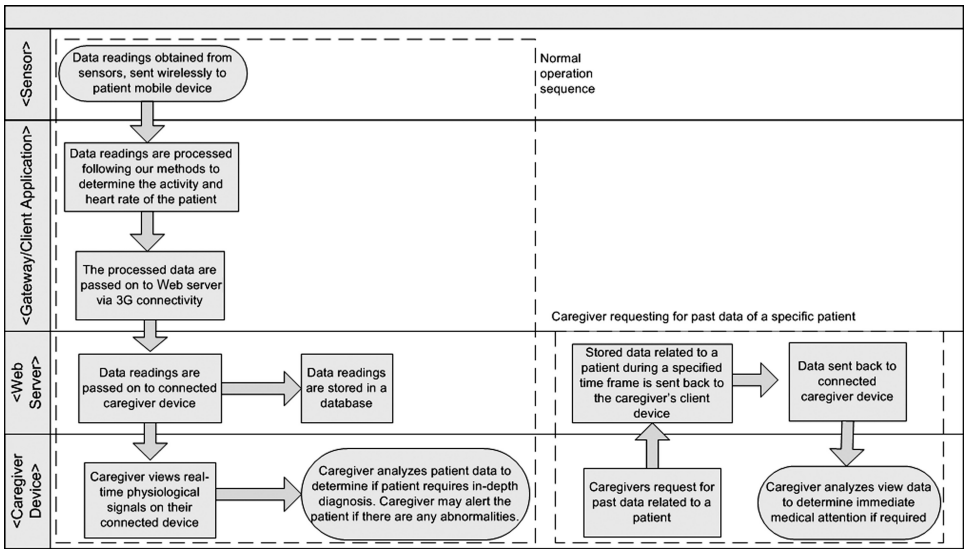


Fig. 3. Ambulatory health monitoring's data flow diagram.

The diagram in Figure 3 shows the interactions among the components of the ambulatory monitoring application (see Figure 2). The data source is obtained from the sensor devices that are attached strategically on the patient's body. Subsequently, these data are sent to the client-side application for real-time processing via a wireless link, which in this case is MobiSense. At MobiSense, these data readings are processed following our methods, which are described in Section 3.3. Upon processing, these data may be transmitted to the central/Web server, which will check if there are requests from the caregiver for real-time monitoring. If it does, the Web server will streamline the data readings to the relevant caregiver. At the same time, the Web server will also store these data in a data repository for future diagnosis. The caregiver may be able to analyze patient data in a real-time as well as off-line manner by requesting the past data from the Web server. Based on the patient's activity and the corresponding heart rate of the patient, the caregiver will be able to determine a life threatening situation. If such a condition is found, the caregiver will alert the patient and emergency services can also be notified to bring the patient to the hospital for further in-depth diagnosis.

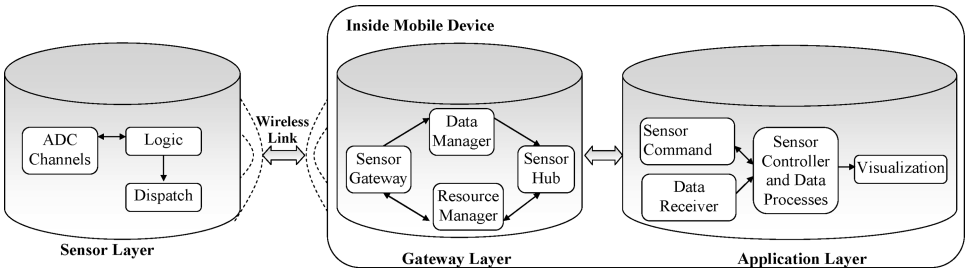


Fig. 4. MobiSense infrastructure.

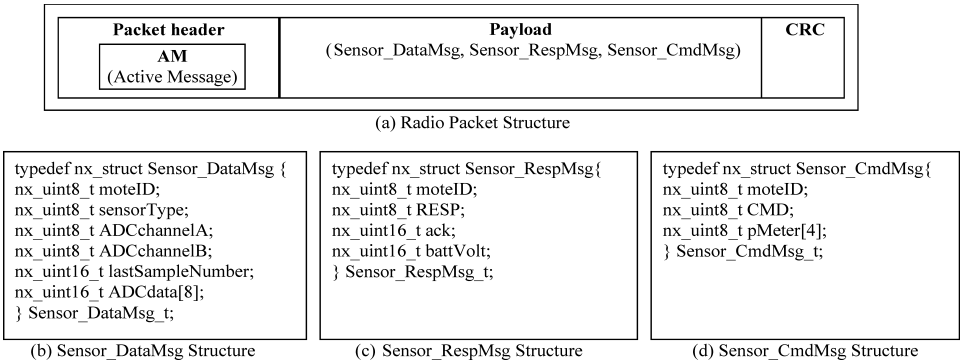
The proposed MobiSense is coded in Java 2 Micro Edition (J2ME) and the corresponding sensor program in each of the sensor nodes is written in nesC [Gay et al. 2003] using the TinyOS 2.x platform. The gateway and sensor programs communicate using a wireless IEEE 802.15.4 (Zigbee) protocol. For the gateway sensor node, we utilize BaseStation, provided by TinyOS 2.x [Berkeley 2009].

Mobisense is comprised of three main layers, namely; *Application*, *Gateway*, and *Sensor* layers. The application layer is concerned with the highest level of the system. This includes visualizations, data processing, and sensor command executions. The gateway layer is located in the middle between the application and sensor layers. This layer serves the incoming and outgoing data from the sensor to the application and vice versa. Both the application and gateway layers reside in the mobile device. The sensor layer deals with sensor node operations, which are attached to the wearer's body. The system infrastructure is depicted in Figure 4. Each layer of the system is further discussed in the following.

3.1 Sensor Layer

This section relates to the sensor node component in the proposed system. It discusses the logic component (see Figure 4) in the sensor layer, which includes packet structure, communication, and various functionalities of the sensor node. The MIG (message interface generator) is required to read the nesC `struct` definitions for message types and generate a java class for each message type. MIG is a tool to automatically generate java classes that correspond to active message types in the sensor node program [Berkeley 2009]. The data type in the sensor layer is comprised of three elements, namely, `Sensor_DataMsg`, `Sensor_CmdMsg`, and `Sensor_RespMsg`. `Sensor_DataMsg` is defined for sensor data readings, `Sensor_CmdMsg` and `Sensor_RespMsg` are for command and response messages, respectively. The generic radio packet structure and the message structure for each of the payloads in the sensor layer is given in Figure 5.

TinyOS applications are comprised of two types of components, namely configurations and modules. Configurations wire components together, while modules provide application code. For simplicity, the nesC functions that we provide in this section refer to the code in the module component. We classify the functions of the sensor layer into three categories: (1) data reading, (2) resource management, and (3) reconfigurations.



(a) Radio Packet Structure

(b) Sensor_DataMsg Structure (c) Sensor_RespMsg Structure (d) Sensor_CmdMsg Structure

Fig. 5. Packet structures.

ADC Data Reading (nesC)

```
/* Set the sampling rate as per default interval */
samplingRate = (1000/DEFAULT_INTERVAL);

event void DataCollectionTimer.fired()
{
/* ADC Counter is incremented and is only read when the */
/* counter is equal to the sampling rate. This function */
/* is applied for bi-axial accelerometers whereby the   */
/* data streams are coming from two channels (x and y)  */

 ADCCounter++;
 if (ADCCounter == samplingRate)
 {
  if (call SensorX.read() != SUCCESS)
     report_problem();
  if (call SensorY.read() != SUCCESS)
     report_problem();
  ADCCounter = 0;
  }
}
```

Fig. 6. Data reading function.

3.1.1 *Data Reading.* Data reading is a typical function in sensor devices. In this function, the sensor data is collected from the ADC channel, which is controlled by the sampling rate. Sampling rate indicates the frequency of data collection, and each data collection is regulated by a timer control. For instance, if the sampling rate is set to 40 Hz, this means the data collection takes place every (1000/40) ms or 25 ms.

These sensor data will subsequently be packed and transmitted over wireless radio to the gateway layer. The gateway layer will read the data packet, and pass on the data to the application layer via the socket. Figure 6 displays the code for data collection in the sensor layer (nesC).

A communication method has been implemented in the sensor gateway as a means to avoid data collision in the context of multiple sensor devices. This method is to ensure that each sensor transmits the data reading in a different time slot. In this case, we deploy a modified polling scheme in the MAC layer, called WBAN MAC. This scheme is courtesy of another project under the EHS-II program, which has been reported in Silva et al. [2008]. This polling scheme adopts a star topology and a master-slave model. In this case the gateway is

the master node and the sensor nodes are slaves. In this method the gateway sends a poll packet to each of the sensor nodes in a round-robin basis. The WBAN MAC differs from standard polling in the sense that each transmission is preceded by a very short carrier sense period, which is applied as a routine check to the channel in order to ensure that the channel is free for transmission each time a packet is to be sent out.

The WBAN MAC provides three communication services: (a) best effort, (b) reliable, and (c) urgent, modes of service. In the Best Effort mode each packet transmission is counted as a successful transmission. Reliable mode adds acknowledgement packets into the process. The MAC replies with an acknowledgment for every packet received. A successful transmission is defined as the sequence of events. The urgent mode differs from the previous two mechanisms in that it does not conform to the poll schedule. Urgent packets are transmitted at the highest available power immediately at the point they are required. This mode is not meant for regular use and is provided for emergency situations. In our system, urgent mode is used for response messages of the resource control commands.

3.1.2 *Resource Management.* Resource management relates to the ability of the system to control and manage each of the sensor nodes in the network. A set of resource control commands is provided in the gateway layer. Applications should initiate the command, which will then be relayed down to the sensor nodes. Resource control messages include sensor once-off and periodic sleep, sensor's battery reading, and notification. Each command from the Gateway layer is received by a designated event in the sensor layer called `event message_t* BaseStationCmdMsgReceive.receive(message_t* msg, void* payload, uint8_t len)` before it is passed on to the relevant function. We do not include this event in the sample codes as it merely forwards the command to the appropriate function.

—*Sensor Once-Off Sleep*. Sensor once-off sleep command refers to the function to turn off the radio communication of the sensor after a predefined duration. This is used to reserve power consumption of the sensor node when the wearer is in low-level of activity (asleep). It was noted that the radio uses orders of magnitude more energy than the microcontroller [Silva et al. 2008]. For instance, an MSP430F1611 microcontroller draws an approximately 1.8 mA current in the active mode and 5.1 μA in sleep mode [Texas Instruments 2009a]. On the other hand, the radio, CC2420 [Texas Instruments 2009b], can consume up to 17.4 mA. Thus, this function helps to minimize the energy utilization of the sensor nodes. Figure 7(a) shows the sensor once-off function in the sensor node.

—*Sensor Toggle*. Sensor toggle function is the command to toggle the sensor node from sleep to wake-up mode and vice versa, continuously following a predefined duration. This function is similar to the sensor once-off command except that this function allows auto-interchange between sleep and wake-up mode in a continuous manner. With this function, the application can send the command once and the sensor radio will toggle on and off until

```
Sleep Radio Once (nesC)
void sleepRadioOnce(uint8_t sleepOnceDuration)
{
 /* This code is used to set the global variable - */
 /* sleepDuration in msec and sends acknowledgment */
 /* to the Gateway layer.                          */
 Sensor_RespMsg_t *dataSection;
 sleepDuration = sleepOnceDuration*1000;

 if (!sendbusy)
  {
  dataSection = (Sensor_RespMsg_t *) call
  SensorRespMsgSend.getPayload(&urgentMessage);
  dataSection->moteID = TOS_NODE_ID;
  dataSection->ack = 13;
   if (call
   BANSend.urgentSend(AM BROADCAST ADDR,&urgentMessage,
   sizeof(Sensor_RespMsg_t)) == SUCCESS)
   { oneShot = TRUE; }
  }
  if (!sendbusy)
    report_problem();
}
event void SensorRespMsgSend.sendDone(message_t* msg,
error_t error) {
 /* This code stops the radio transmission in a    */
 /* certain duration.                              */
  if(oneShot)
  {call BANRadio.radioOff();
   call OneShotTimer.startOneShot(sleepDuration);}
}

event void OneShotTimer.fired() {
 /* This code re-starts the radio transmission.    */
call  BANRadio.radioOn();
}
```

(a) Sensor Once-Off Sleep Function

```
Sleep Radio Toggle (nesC)
void sleepRadioToggle(uint8_t sleepToggleDuration)
 {
 /* This code is used to set the global variable -      */
 /* sleepDuration in msec and sends acknowledgment      */
 /* to the Gateway layer.                               */
 Sensor_RespMsg_t *dataSection;
 sleepDuration = sleepToggleDuration*1000;
 if (!sendbusy)
 {
 dataSection = (Sensor_RespMsg_t *)
 call SensorRespMsgSend.getPayload(&urgentMessage);
 dataSection->moteID = TOS_NODE_ID;
 dataSection->ack = 14;
    if (call
    BANSend.urgentSend(AM_BROADCAST_ADDR,&urgentMessage,
    sizeof(Sensor_RespMsg_t)) == SUCCESS)
    {toggling = TRUE;}
  }
 if (!sendbusy)
   report_problem();
}
event void SensorRespMsgSend.sendDone(message_t* msg,
error_t error) {
 /* This code sets the toggle radio parameter           */
 /* based on the sleepDuration interval.                */
 if(toggling)
  {radioOff = TRUE;
   call BANRadio.radioOff();
   call ToggleTimer.startPeriodic(sleepDuration);}}

event void ToggleTimer.fired() {
/* This code toggles the radio transmission on/off.     */
 if(radioOff)
   call BANRadio.radioOn();
 if(!radioOff)
   call BANRadio.radioOff();
 if(radioOff)
    radioOff = FALSE;
 else
    radioOff = TRUE; }
```

(b) Sensor Toggle Function

Fig. 7. Sensor once-off and toggle functions.

the application sends out a cancel toggle command. Figure 7(b) depicts the sensor toggle function.

—*Check Battery Level and Notification*. The Sensor layer is also equipped with a function to check the battery level of each sensor node in a regular interval. When the gateway layer detects one or more sensor nodes with battery level below a certain threshold, it will notify the application accordingly and the application may display the battery level to the application user. The gateway layer is to alert the user of the application when the battery level of a certain sensor is below the predefined threshold. The corresponding function is depicted in Figure 8.

All resource management functions are designed to control either a specific sensor or a group of the same type of sensors (accelerometers, ECG, etc.). When there are various types of sensors being deployed, it is easier for the application to call the group sensor commands as it helps to ensure that each of these sensors conforms to the same setting.

3.1.3 *Reconfiguration.* Reconfiguration feature enables applications to change the property of sensor nodes on-the-fly, such as changing the sampling rate of the sensor through the gateway layer. Similar to the resource management functions, reconfiguration also supports settings or changes based on the individual node ID or a group of the same sensors. In the case of a critical self-wake feature whereby the sensor node is set to sense critical data during sleep-mode, the application is able to define and modify the critical value parameters in the sensor nodes. Each of the reconfiguration schemes is described in the following.

Get Battery Voltage (nesC)

```
void getBatteryVoltageLevel() {
/* It reads the current battery level.                        */

 if (call Battery.read() != SUCCESS)
      report_problem();
 }

event void Battery.readDone(error_t result, uint16_t data) {
/* It sends out the current battery level to Gateway layer  */

  Sensor_RespMsg_t *dataSection;
  dataSection = (Sensor_RespMsg_t *) call
   SensorRespMsgSend.getPayload(&urgentMessage);
   dataSection->moteID = TOS_NODE_ID;
   dataSection->battVolt = data;

   if (call
   BANSend.urgentSend(AM_BROADCAST_ADDR,&urgentMessage,
   sizeof(Sensor_RespMsg_t)) == SUCCESS){
   sendbusy = TRUE;
   }
   else
   report_problem();
}
```

Fig. 8. Check battery level and notification function.

Get Sampling Rate (nesC)

```
void getSamplingRate()
{
 /* The following code is used to return the        */
 /* current sampling rate in HZ.                    */

 Sensor_RespMsg_t *dataSection;
 dataSection = (Sensor_RespMsg_t *)
 call SensorRespMsgSend.getPayload(&urgentMessage);
 dataSection->moteID = TOS_NODE_ID;
 dataSection->RESP = (1000/samplingRate);
 if (call
  BANSend.urgentSend(AM_BROADCAST_ADDR,&urgentMessage,
  sizeof(Sensor_RespMsg_t)) == SUCCESS)
 {
   sendbusy = TRUE;
 }
}
```

(a)Get sampling rate function

Set Sampling Rate (nesC)

```
void setSamplingRate(uint8_t newSamplingRate)
{
 /* The following code sets the sampling rate to the  */
 /* new rate as specified and sends an acknowledgment */
 /* to the Gateway layer.                             */

  Sensor_RespMsg_t *dataSection;
  samplingRate = (1000/newSamplingRate);
  if (!sendbusy)
  {
  dataSection = (Sensor_RespMsg_t *) call
  SensorRespMsgSend.getPayload(&urgentMessage);
  dataSection->moteID = TOS_NODE_ID;
  dataSection->ack = 12;
   if (call
   BANSend.urgentSend(AM_BROADCAST_ADDR,&urgentMessage,
   sizeof(Sensor_RespMsg_t)) == SUCCESS)
    {
    sendbusy = TRUE;
    }
  }
 else
  report_problem();
 }
```

(b)Set sampling rate function

Fig. 9. Get and set sampling rate function.

—*Get Sampling Rate*. Get sampling rate function provides the ability for the application to check the current sampling rate of the sensor node. Figure 9(a) shows the get a sampling rate function in the sensor layer.

—*Set Sampling Rate*. This set sampling rate function enables the application to modify the sampling rate of a certain sensor node or a group of sensor nodes. In the case of group commands, the application simply specifies the command and the target sensor type, and the sampling rates of each of the sensors that belong to that group will be updated accordingly. The set

sampling rate function, which is shown in Figure 9(b), is similar to the get-sampling rate except that this function receives a parameter value for the new sampling rate for a defined sensor node/group.

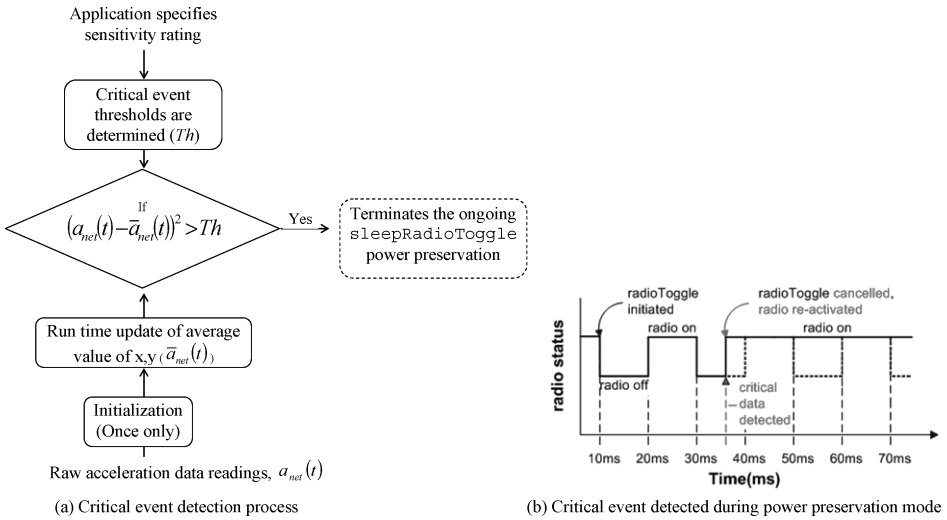
—*Critical Self-Wake.* Critical self-wake is a feature to allow sensors to conserve battery power without compromising on the potential loss of critical data, should a critical event occur while the sensor node's transceiver is inactive. This ensures that the application will still be informed of any critical condition experienced by the patient while exercising sensor node power preservation. In this function, the application is required to specify a percentage value that is used to calculate the equivalent threshold in order to indicate to sensor nodes which data value(s) should be considered critical. Upon detecting the occurrence of critical data (threshold is exceeded), the sensor node shall then change its transceiver's status from inactive to the active mode and resume its normal transmission state—sending the critical data along with all subsequent data readings to the gateway layer, which is responsible for passing all data that it receives onwards to the application.

The critical threshold is set by the application and it can be dynamically changed at run-time in accordance to the current activity of the user. For instance, if the user were sitting, the critical wake threshold would be different from when he/she were lying down. The gateway layer allows the application to specify its desired threshold in terms of a percentage number. The higher the value of this percentage, the more sensitive the sensor is to the movement of the user. For example, a high setting (e.g., 80% and above) means that the sensor will signal a critical event occurrence when a small amount of force (or slight movement) is exerted on the accelerometer sensor.

A run-time update of the average net acceleration ($\bar{a}_{net}(t)$) is performed each time the accelerometer sensor detects new pair biaxial data (x and y axis), $a_{net}(t)$. The average net acceleration is calculated by adding the square of each of the new data readings $a_{net}(t)$ to $\bar{a}_{net}(t)$ and dividing based on the length of computation. Besides the run-time updating of the average net acceleration, the fluctuation, $(a_{net}(t) - \bar{a}_{net}(t))^2$, of the new pair of x and y axis data from the average net acceleration value is also compared against the critical threshold, Th, to determine if the threshold is exceeded. Threshold values are obtained based on an empirical mapping of the forces of different magnitude (no movement, slight movement, vigorous movement) as applied to the accelerometer sensor. An overview of the critical self-wake feature is illustrated in Figure 10. Figure 10(a) shows the critical event detection process, and Figure 10(b) depicts the situation whereby the critical event is detected during power preservation, and the radio transceiver is reactivated to resume data transmission.

3.2 Gateway Layer

The gateway architecture is illustrated in Figure 11. As shown in Figure 11, the sensor nodes transmit data to the gateway layer. The gateway layer



(a) Critical event detection process

(b) Critical event detected during power preservation mode

Fig. 10. Critical self-wake feature.

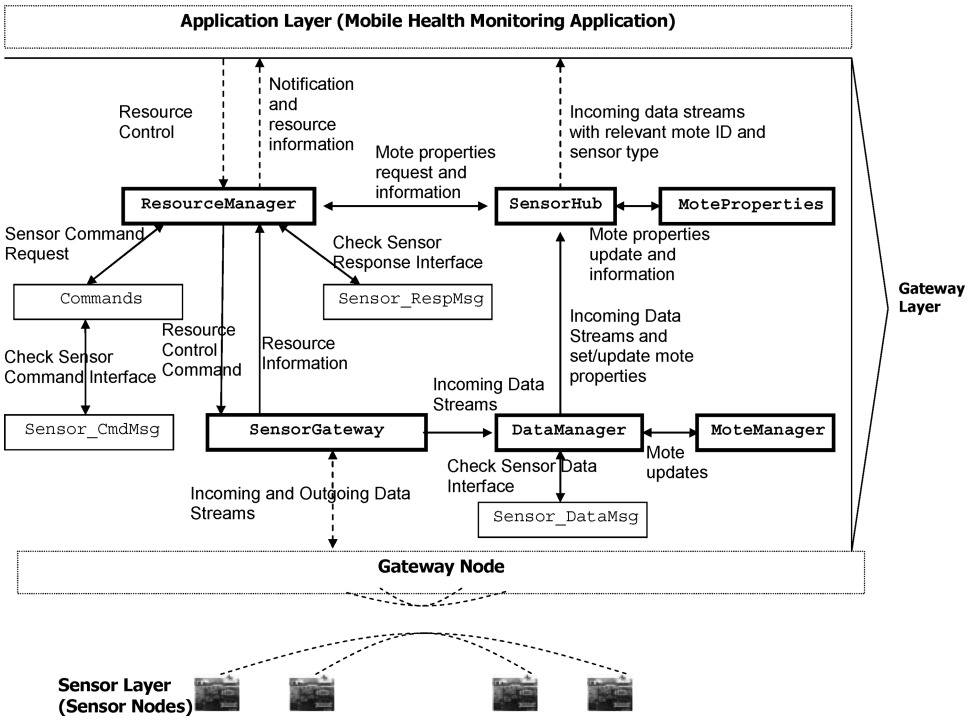


Fig. 11. Gateway architecture.

processes the data and passes them on to the application layer. This layer contains six main classes, (1) SensorGateway, (2) DataManager, (3) MoteManager, (4) SensorHub, (5) MoteProperties, and (6) Resource Manager. Each of these classes is described in the following.

SensorGateway is the class that initiates the connection to the gateway mote by directly connecting to the communication (COM) port. This class receives raw data streams from the sensor nodes and sends command messages down to the sensor layer. Each sensor is therefore required to have a uniform packet length and structure. SensorGateway is initialized by SensorHub and holds an instance of DataManager and of ResourceManager. SensorGateway processes the incoming data streams from the sensor nodes by checking the type of inbound packet. If the data contains sensor readings, SensorGateway forwards the data to DataManager, should the payload contain response messages, they are forwarded to ResourceManager.

Similarly, SensorGateway also receives command messages from ResourceManager and sends the commands directly to the sensor nodes via the COM port to the attached gateway nodes, which then transmit the command out wirelessly. This class also implements both the Sensor_DataMsg and the Sensor_RespMsg interfaces.

DataManager receives data packets from SensorGateway. It strips off the packet information (channel number, source node ID, SensorType) from the data message itself. Subsequently, it forwards the packet information to the MoteManager in order to detect if there is any lost node or node that is turned off within the network. MoteManager is responsible for managing the sensor node. It detects node and channel information such as how many nodes are currently active, the node and channel ID, packet number and so on. This class is applied for sensor node monitoring and corresponds to the *plug-and-play* function of the system.

DataManager is also responsible for storing and updating packet information in the MoteProperties class. Therefore, when MoteManager detects a channel loss, it passes the information about the loss to DataManager, which will update MoteProperties accordingly. MoteProperties is predominantly used to store and maintain each sensor's properties. This class particularly interacts with ResourceManager and DataManager through SensorHub.

SensorHub is the only class with a main method in the architecture. It functions as an initiator and connector between classes. This class also defines the sensor platform and baud rate upon initialization. As SensorGateway is a root class, which receives and sends data to and from the sensor node, two instances of SensorGateway are created. One instance is for receiving the data streams, and the other is for sending commands to the sensor node. By connecting this class with the application, the user is able to observe the signals of the BSN sensor, perform signal processing on them, send downstream commands to the sensor, and receive resource information.

ResourceManager is mostly concerned with resource control and management of the sensor nodes. It has three main functions: (1) sensor toggle/once-off, (2) power monitoring, and (3) sampling rate check and adjustment. The list of applicable command messages is available in the Commands class.

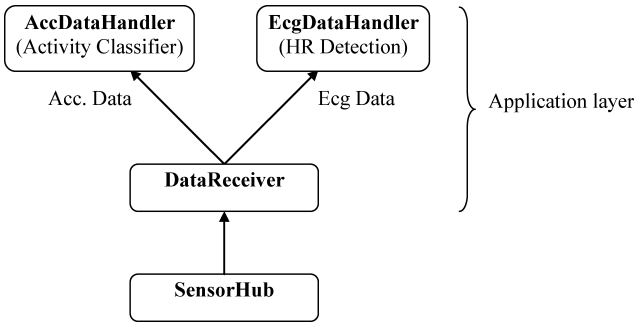


Fig. 12. Data flow diagram in the application layer.

`ResourceManager` mostly interacts with `MoteProperties` in order to read and adjust the sensor's properties, and `SensorGateway` to send commands and receive responses accordingly. This class implements the `Sensor_CmdMsg` interface. Because of the possibility of missing commands, reinforcement schemes to ensure that each command is received by the sensor node, have also been implemented in this class. These schemes are not only useful for tracking each command but also to avoid application layer resending the command, which subsequently minimizes communication overhead between these two layers.

3.3 Application Layer

This section discusses the processing of sensor data received from the sensor layer through the gateway layer. `DataReceiver` in the application layer is the initial class that receives sensor readings from `SensorHub` in the gateway layer (see Figure 11). `DataReceiver` reads each data packet and segregates the data based on the type of sensor from which the data originates. For accelerometer data, `DataReceiver` forwards the sensor readings to the `AccDataHandler`. Similarly, ECG data are passed on to the `EcgDataHandler` for further processing. These two classes, `AccDataHandler` and `ECGDataHandler` relate to the logic of the application layer. Figure 12 depicts the data flow diagram of the application layer. `AccDataHandler`, which is associated with our activity classifier scheme, is discussed in Section 3.3.1. It is then followed by `ECGDataHandler`, which describes our heart rate detection scheme in Section 3.3.2.

3.3.1 *Activity Classifier.* In order to support activity classification, accurate estimation of a patient's flexion angles is important, and a dynamic filtering technique is one of the most suitable approaches to consider [Dong et al. 2006]. Our activity classifier is carried out following our ruled-based heuristic activity classification scheme, which utilizes an *EK* filtering algorithm for tracking the flexion angles of the human body. The classification scheme is comprised of *Postural* and *Movement Classifiers* for activity tracking.

As shown in Figure 13, the raw accelerometer sensors data is processed by the EK filters, *EKFtrunk* and *EKFgait*, to estimate the trunk and thigh flexion angles, respectively. The derived angles are required by the *Classifier Selector*, and are subsequently passed on to the *Postural* and *Movement Classifiers*.

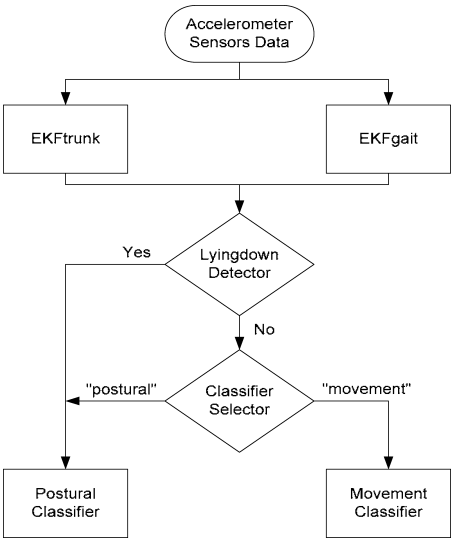


Fig. 13. Flowchart diagram of our rule-based heuristic scheme.

The *Lyingdown Detector* is to provide a quick indication of the lying down posture if the trunk sensor's vertical acceleration is less than 0.98 m/s^2 while the horizontal acceleration is greater than 8.82 m/s^2 or less than 0.98 m/s^2. These thresholds are determined empirically by placing the sensors at a lying position. The *Classifier Selector* may be passed over if the sensor wearer is indeed in the lying down posture. Each of the components involved in our activity classification scheme is described in the following.

—*EK Filtering Algorithm.* The EK filter is employed to provide an accurate estimation of the flexion angles of the human body. In our classification system, the filter is located in *EKFgait* as well as *EKFtrunk* (by setting the parameter for the radius of rotation to 0). This filtering algorithm obtains the discrete-time dynamics by discretizing the continuous-time kinematic model that adopts the *Continuous Wiener Process Acceleration* (CWPA) model whereby disturbances in the system dynamics are modeled as random inputs. In the CWPA model, the motion of a constant angular acceleration object for the flexion angle θ is described by the, equation $\ddot{\theta} = \tilde{v}$ where $\tilde{v}$ is a continuous-time zero-mean white noise process, which models the changes of the acceleration. Thus, the continuous-time state equation is:

$$\dot{\mathbf{x}} = \mathbf{F}\,\mathbf{x} + \mathbf{D}\tilde{v}, \tag{1}$$

where,

$$\mathbf{x} = \begin{bmatrix} \theta \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{2}$$

This is a third-order state model with three integrations. The corresponding discrete-time state equation with sampling period T is:

$$\mathbf{x}_k = \mathbf{A}\,\mathbf{x}_{k-1} + \mathbf{v}_{k-1}, \tag{3}$$

where $\mathbf{A}$ is the transition matrix:

$$\begin{aligned}\mathbf{A} &= e^{FT} \\ &= \mathrm{L}^{-1}[(s\mathrm{I} - \mathrm{F})^{-1}] \\ &= \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix},\end{aligned} \tag{4}$$

and

$$\mathbf{v}_k = \int_0^T e^{\mathbf{F}(T-\tau)}\,\mathbf{D}\tilde{v}_{kT+\tau}\,d\tau. \tag{5}$$

The covariance matrix of the zero-mean $\mathbf{v}_k$ is derived from:

$$\begin{aligned}\mathbf{Q} &= \mathrm{E}\left\{\mathbf{v}_k\mathbf{v}_k'\right\} \\ &= \begin{bmatrix} \frac{T^5}{20} & \frac{T^4}{8} & \frac{T^3}{6} \\ \frac{T^4}{8} & \frac{T^3}{3} & \frac{T^2}{2} \\ \frac{T^3}{6} & \frac{T^2}{2} & T \end{bmatrix} \underset{\sim}{q},\end{aligned} \tag{6}$$

where

$$\underset{\sim}{q} = \mathrm{E}\left\{\tilde{v}_{kT+\tau}\ \tilde{v}_{kT+\tau}\right\}. \tag{7}$$

The time updates of the state estimate and estimation-error covariance are defined as follows:

$$\begin{aligned}\hat{\mathbf{x}}_k^- &= \mathbf{A}\hat{\mathbf{x}}_{k-1}^+ \\ \mathbf{P}_k^- &= \mathbf{A}\,\mathbf{P}_{k-1}^+\mathbf{A}^{\mathrm{T}} + \mathbf{Q}\end{aligned}. \tag{8}$$

Figure 14 illustrates the body dynamics that have been adopted into the system model of the EK filter. Besides the presence of the gravitational acceleration, which every object on earth experiences, we incorporate another two acceleration components for the description of body segments dynamics. When the body segment swings with constant speed, the swing motion can be described by a radial acceleration component, that is directed toward the center of the motion. Moreover, there exists another tangential acceleration component, which can model the change in speed of the body segment, and these two acceleration components are always perpendicular to each other [Giancoli 1998]. In brief, the three acceleration components that describe this model are the gravitational acceleration, radial acceleration, and tangential acceleration.

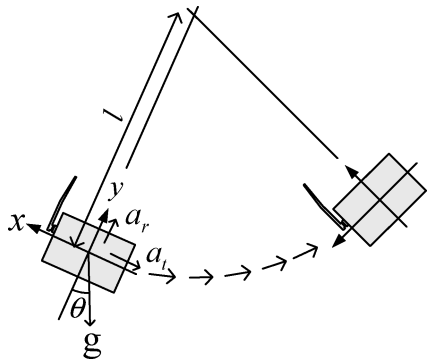


Fig. 14. Classifier system model: an illustration.

Using the classifier model in Figure 14, and the variable definition as illustrated, the output measurement of the accelerometer sensor can be defined as follows:

$$\begin{aligned} &\text{y - reading} : y_k = a_{r,k} - \mathrm{g}\cos(\theta_k) + n_{y,k} \\ &\text{x - reading} : x_k = -a_{t,k} - \mathrm{g}\sin(\theta_k) + n_{x,k} \end{aligned} . \tag{9}$$

Let g be the earth gravity. $a_{r,k} = \dot{\theta}_k^2 l$ and $a_{t,k} = \ddot{\theta}_k l$ are the discrete-time measurements of the *tangential* and *radial* acceleration components, respectively. The measurement noise, $n_{y,k}$ and $n_{x,k}$, are additive, zero mean, white, and uncorrelated with the process noise. Equation (9) can be rewritten as:

$$\mathbf{y}_k = \mathbf{h}_k(\theta_k, \dot{\theta}_k, \ddot{\theta}_k) + \mathbf{n}_k. \tag{10}$$

Given Equation (10), the measurement update for the first-order EK filter is derived based on a first order Taylor series expansion of the nonlinear measurement equations around $\mathbf{x}_k = \hat{\mathbf{x}}_k^-$. The corresponding updated state error covariance can be derived from:

$$\begin{aligned} \hat{\mathbf{x}}_k^+ &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left[\mathbf{y}_k - \mathbf{h}_k\left(\hat{\theta}_k^-, \hat{\dot{\theta}}_k^-, \hat{\ddot{\theta}}_k^-\right)\right] \\ \mathbf{P}_k^+ &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{R}_k \end{aligned} . \tag{11}$$

Let $\mathbf{R}_k$ be the covariance matrix of $\mathbf{n}_k$, and $\mathbf{K}_k$ the EK filter gain. Then the following equation can be applied:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\,\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k)^{-1}, \tag{12}$$

and $\mathbf{H}_k$ is the partial derivative matrix that can be obtained from:

$$\begin{aligned} \mathbf{H}_k &= \left.\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}\right|_{\hat{\mathbf{x}}_k^-} \\ &= \begin{bmatrix} \mathrm{g}\sin(\hat{\theta}_k^-) & 2\,\hat{\dot{\theta}}_k^-\,l & 0 \\ -\mathrm{g}\cos(\hat{\theta}_k^-) & 0 & -l \end{bmatrix}. \end{aligned} \tag{13}$$

—*Classifier Selector*. This classifier determines if the underlying physical activity belongs to either the *postural* or *movement* class, based on the state diagram illustrated in Figure 15. In Figure 15, P and S denote the periodic and static gait activities, respectively. The average value of the minimum

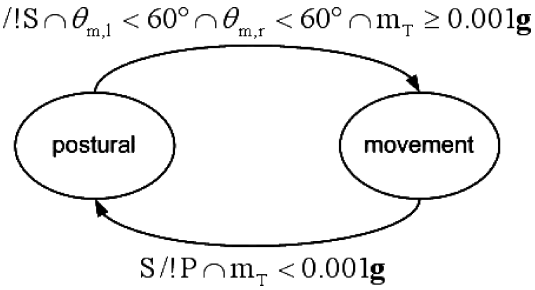


Fig. 15. Method for determining the type of physical activity performed by the wearer.

and maximum flexion angles falling within the time duration of 0.5 s are denoted by $\theta_{m,l}$ and $\theta_{m,r}$ for both the left and right thighs respectively. m_T is the variance of the trunk sensor's vertical acceleration, and$P/!S$ denotes a state where the previous gait is non-static whereas the current gait is periodic. A periodic gait activity implies that both thighs are swinging in the opposite direction within the specified time duration.

—*Postural Classifier*. The *Postural Classifier* provides output as to whether the underlying posture is lying down, sitting, or standing, based on the set of trunk (θ_{tr}), left thigh (θ_l), and right thigh (θ_r) flexion angles. For *lying down* posture classification, all angles are to fall within the range of 60° to 120°. For *sitting* posture, θ_{tr} is to fall within 15° and 55° while both thigh angles are to fall within 35° and 160°. As for the *standing* posture, all angles are to be within the specified range of −30° and 30°.

—*Movement Classifier*. The *Movement Classifier* is applied to determine the underlying activities as either walking, jogging, or running, by estimating the walking speed. The estimation method is derived from the fact that the force exerted by an object is directly proportional to the acceleration, and thus physical activity intensity will be a function of acceleration.

With $a_{L,net}$ and $a_{R,net}$ representing the net acceleration of the left and right thighs, we define the *average net acceleration* (ANA) as the time average obtained from the summation of $a_{L,net}$ and $a_{R,net}$ over a time duration of 2 s. Figure 16 depicts a typical ANA histogram distribution of two walking speeds. The proposed walking speed estimator makes use of the least squares (LS) approach to fit the mean value of the ANA, x, to the third-order polynomial signal model. From this, we obtain the estimated walking speed $y = -0.1094 + 4.3871x - 0.9508x^2 + 0.1373x^3$, as illustrated in Figure 17. The bold line in Figure 17 indicates the experimental data fitting using LS.

3.3.2 *Heart Rate Detection Scheme.* This section briefly describes our heart rate (HR) detection scheme. Figure 18 depicts the flow diagram of the ECG data processing. `ECGDataHandler` contains a new data arrival event, which is invoked when three new ECG data packets (consisting of 24 data values) arrive. These data are forwarded to `ECGRenderer` as well as to `QRS_Detector`, which display the ECG waveform and determine the R-peak position from these data, respectively.

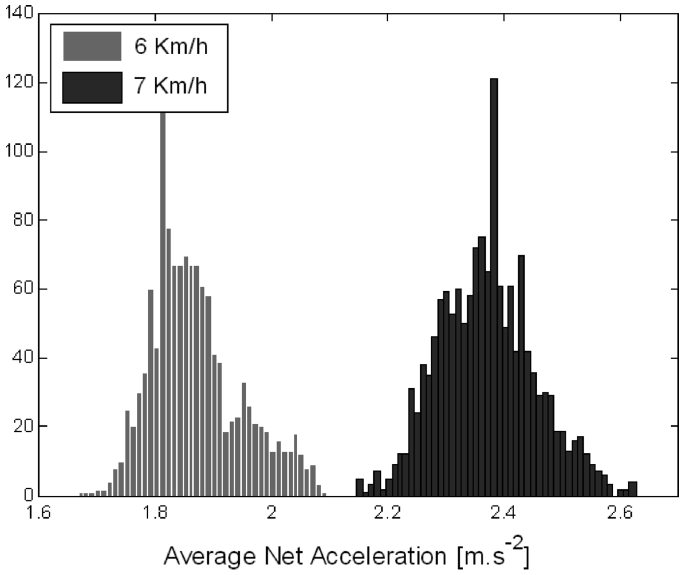


Fig. 16. ANA histogram plot of walking on a treadmill at 6 km/h and 7 km/h.

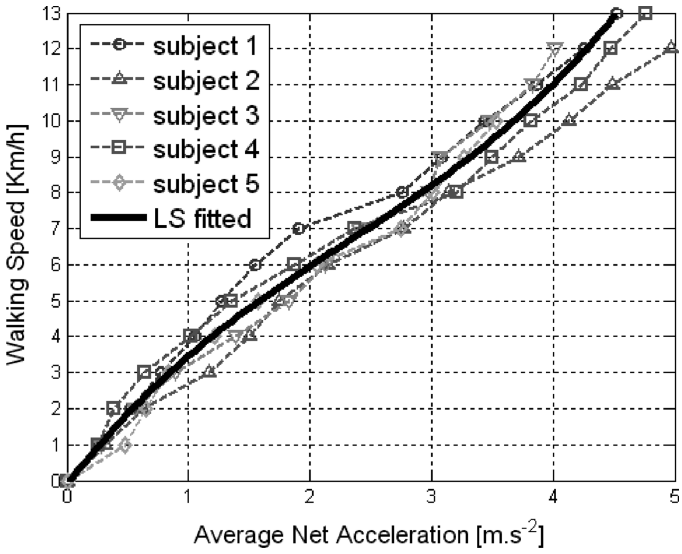


Fig. 17. Walking speed against the net acceleration averaged over a time duration of 2 s for five test subjects.

`ECGRenderer` returns QRS position and QRS trace-back position, which are needed to determine the position to draw the peak line on the ECG waveform, and to calculate the R-R interval. The R-R interval is subsequently utilized to determine the heart rate value by `HR_Measurement`. The heart rate value is measured using Equation (14) and the value is then displayed on the

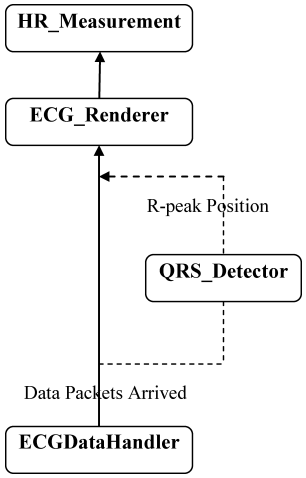


Fig. 18. ECG data flow diagram.

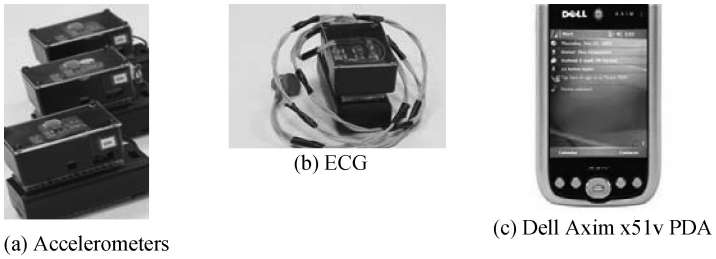


Fig. 19. Sensor and mobile devices used in the prototype.

application.

$$HR = \left(\frac{60000}{RR - \text{interval}}\right) beats/\min . \tag{14}$$

4. TEST BED

This test bed is made up of two parts; (1) to show the prototype of our proposed MobiSense, and (2) to evaluate the performance of the system. As for the sensor devices, we use three Imperial College BSN accelerometers and one ECG sensor [Lo et al. 2005; Lo and Yang 2005]. This BSN sensor platform is mainly utilized for research purposes, and its attractiveness lies in the small size of the sensors—about 26 mm which is relatively smaller than other platforms on the market. Our system resides in the PDA (Dell Axim x51v). Figure 19 shows the sensor nodes and the mobile device for the prototype.

4.1 Prototype System

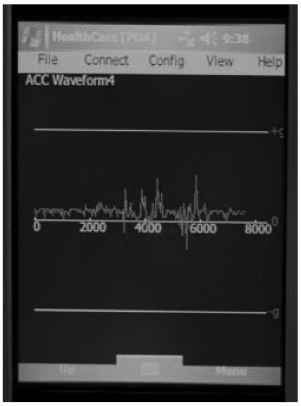
We describe a of MobiSense deployment in a real-world environment. Three wearable wireless accelerometer sensors are attached to the trunk and both thighs of a human subject. The sampling rate for the accelerometer sensor is

(a) One accelerometer and ECG sensor worn on the belt

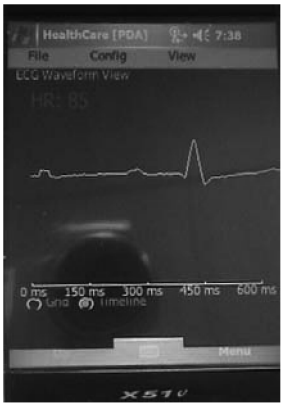
(b) One accelerometer sensor worn on the right-leg

(c) One accelerometer sensor worn on the left-leg

Fig. 20. Sensor placement on the human's body.



(a). Accelerometer waveform



(b). ECG waveform

Fig. 21. Accelerometer and ECG waveform.

set to 25 Hz. One ECG sensor is also attached to the chest to detect the heart rate of the person, and its sampling rate is set to 100 Hz. Figure 20 shows the placement of these wearable sensors on the human's body.

The basic function to demonstrate is data acquisition. In this case, the system receives the data readings from the four active sensor nodes and displays them to the user. In Figures 21(a) and (b), we show an accelerometer and ECG readings that have been rendered by the system, respectively.

In Figures 22(a–d), we present a sequence of activities ranging from sitting, lying down, standing, and walking. These activities are being tracked by our classification scheme in displayed on real time. As depicted in Figure 22, our

(a) Sitting activity (b) Lying-down activity

(c) Standing activity (d) Walking activity

Fig. 22. Performing activities and tracking.

systems, with built-in human animation, is able to follow and mimic these activities accordingly, and all of these are Adisplayed on in the user's screen. The output of the activity classification together with the patient's heart rate may be stored locally or transmitted via wireless network to a central server for further processing, which is beyond the scope of this article.

4.2 System Evaluation

This section evaluates the system performance, including power consumption of the sensor nodes, real-time and query latency as well as the critical-self-wake detection method. Furthermore, we have also carried out experiments with human subjects to investigate the accuracy of our classifier algorithms.

4.2.1 *Power Consumption, Real Time and Query Latency Performance.* We present the performance of the proposed system based on the prototype that has been described in Section 4.1. The performance measurement includes, (1) power consumption of the sensor the node, (2) real-time, and (3) query latency. The parameters of concern are defined in Table II.

—*Power Consumption.* This case is concerned with investigations of the sensor node's power consumption when sensorToggle operation is activated and deactivated. In our experiments, the sensor node is using a 2xAA battery model. Figure 23 the wattage level when executing sensorToggle and without

Table II. Parameters of Concern

Parameters	Value
Wireless link bandwidth	250 kbps
Data Size (Payload)	24 bytes
Query Size	6 bytes
Response Message Size	6 bytes
Number of Data Packet	Up to 200

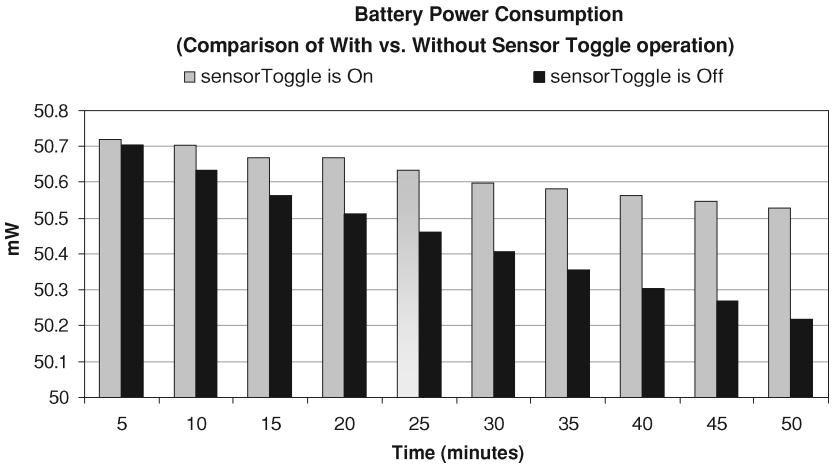


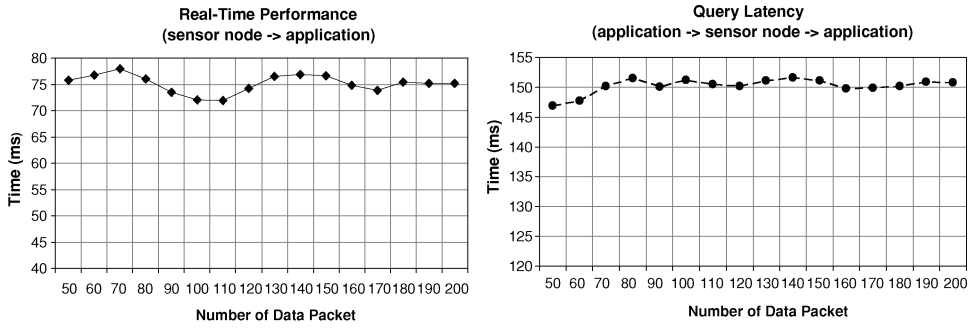
Fig. 23. Battery power consumption with and without sensorToggle operation.

sensorToggle. It indicates that sensorToggle incurs substantially lower power consumption and hence conserves a significant amount of battery power.

Figure 23 also indicates that the discrepancy in power consumption between the two operations becomes larger over time due to the significant difference in the consumption rates. Power preservation schemes like sensorToggle are able to prolong the life-time of the sensors, which helps to reduce the number of times patients have to replace batteries. This is significant because some patients like the ill and elderly may have difficulty in changing battery of the sensor.

—*Real-Time and Query Latency Performance.* Real-time performance is of much importance in the healthcare applications as it involves critical sensor readings of the patient's health conditions. Therefore, we measure the performance of the proposed system in terms of the amount of time in which the sensor data arrives at the application layer.

As shown in Figure 24(a), the system is able to receive sensor data within 75 ms on average. Considering video data like in the homecare scenario, which is prone to issues such as video quality, throughput, delay, delay jitter, and so on, can have a maximum delay of 250 ms [Negi and Rajeswaran 2004], we can achieve the same result with considerably better real-time performance.



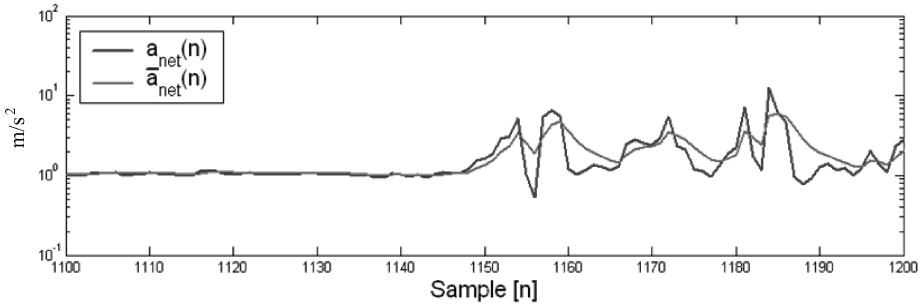
(a) Real-time (b) Query latency

Fig. 24. Real-time and query latency performance.

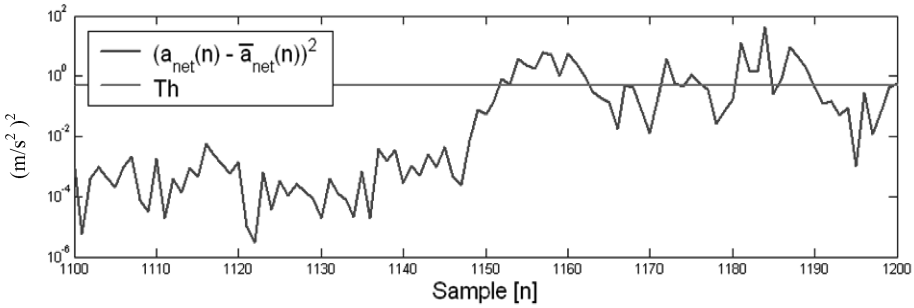
To obtain the response performance of the system, we need to measure the query latency. Query latency is the time required by the application to send a request to the sensor layer and receive the results. This is particularly relevant to the sensor reconfiguration and resource management features of the system. As a result of four iterative experiments, we can observe from Figure 24(b) that the query latency performance of the system varies between 140 and 150 ms. It is reasonable that the major portion of the resulting time was due to the fact that the sensor data and request has to undergo back-to-back cross-platform trips between sensor (nesC/TinyOS) and gateway (J2ME) before the result is received.

4.2.2 *Critical-Self-Wake.* This section describes an experiment regarding the critical-self-wake feature in our system. This feature is described in Section 3.1.3. The aim of this feature is to enable accelerometer sensors to detect motion while exercising sensor node power preservation. In our system, sensors are automatically put into a power preservation mode. Its associated critical-self-wake feature is enabled by the application when a particular activity has been on-going for a certain period of time (e.g., sitting for 30 minutes). This is particularly important since energy in a wireless sensor network is a very precious resource, while on the other hand, health monitoring applications cannot afford to miss vital data that may indicate a patient's critical condition. In this experiment, a sensor was initially placed in a static position; increasing amounts of force were applied to, the sensor in order to simulate a real-life situation in which this feature will be useful, such as when the patient rests on the bed.

When a person rests on the bed, minor movements are normal as patients typically turn about during their sleep. These should not be detected as a critical event and the sensor node should remain in its power-preservation mode. Only when a large movement is detected, with a force that is of a sufficient magnitude to exceed the threshold established by the application, should the accelerometer sensor deactivate its ongoing power preservation and resume the transmission of its sampled data readings. This implementation allows applications to make the final diagnosis and decide if the patient requires emergency assistance.



(a) Raw accelerometers data and average net acceleration



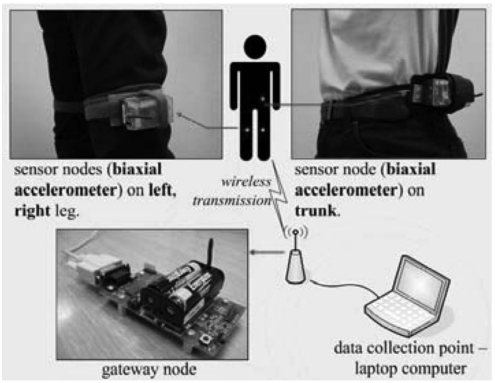
(b) Check against threshold

Fig. 25. Critical-self-wake experiment.

Large movements may relate to a situation in which the patient falls from the bed onto the floor, a potentially life-threatening incident, especially for elderly folk.

Figure 25 depicts the force measurements over 1200 samples data. Figure 25(a) shows the $a_{net}(t)$ and $\bar{a}_{net}(t)$ from the sample data, and Figure 25(b) compares the results with the threshold value (Th) set by the application. Each spike in Figure 25(a) corresponds to a vigorous movement of the accelerometer sensor. In Figure 25(b), minor movements which result in small deviations do not exceed the threshold, whereas vigorous movements, where deviations increase when the accelerometer is moved with a larger amount of force are shown to exceed the threshold. This comparison enables us to determine if the movement of the patient is minor or otherwise. When vigorous movement is detected and its value exceeds the threshold defined by the application, the sensor will resume normal data transmission and start transmitting the raw accelerometer readings to the application for further verification.

4.2.3 *Activity Tracking and Classification.* This section analyzes the performance of our activity classifier algorithms. In order to obtain more reliable measurements, we deploy a commercial sensor platform, Crossbow MicaZ [Crossbow technology 2009]. The setup for the experiment is shown in Figure 26(a).



(a) System set-up for the test

(b) Activities for classification test

Fig. 26. System set-up and various activities.

As depicted in Figure 26(a), three wireless accelerometer sensors, comprised of a Crossbow MicaZ mote processor radio platform (MPR2400) and a Crossbow MicaZ biaxial accelerometer sensor board (MTS310CA), are attached to the trunk and both thighs of a human subject. The sampling rate for the accelerometer sensor is set to 25 Hz. A MicaZ node attached to on MIB510 serial interface board functions as a base-station. The received data is aggregated and processed on a laptop via its RS-232 interface.

Two pilot studies have been conducted with five test subjects. The first study consisted of a series of 40 tasks randomly selected by the evaluator from a set of lying, sitting, standing, and walking activities. Test subjects executed these activities accordingly, as shown in Figure 26(b). Transitional activity is accepted if the activity is a logical transition between two activities, for example, sitting is accepted as a transitional activity from lying to standing. No misclassification was detected throughout this study.

The second study is conducted on five test subjects running on a treadmill at speeds progressively increasing from 1 km/h to 13 km/h, depending on their physical fitness. The results of their estimated walking speeds are tabulated in Table III. This study gave an overall mean-square error (MSE) of 0.91 $(\text{km/h})^2$. The removal of a few bad samples with abnormally large values, such as that shown by subject 4 at 3 km/h, results in a significant improvement of the overall MSE.

The MSE is computed using the equation $MSE_i = \frac{1}{N_i}\sum_{j=1}^{N_i}(x_{i,j} - \bar{x}_i)^2$, where the subscript i indicates a particular test set, $x_{i,j}$ and $\bar{x}_i$ corresponds to the estimated and true values. For simplicity, let us assume that there are two tests of recorded lengths, N_1 and N_2, respectively. Thus, we have $MSE_1 = \frac{1}{N_1}\sum_{j=1}^{N_1}(x_{1,j} - \bar{x}_1)^2$, $MSE_2 = \frac{1}{N_2}\sum_{j=1}^{N_2}(x_{2,j} - \bar{x}_2)^2$and so on. The overall MSE

Table III. Results of Experiments

Walking Speed (km/h)	Test subject 1	2	3	4	5	Overall MSE
1	0.04	0.16	0.01	0.09	0.68	0.20
2	0.03	0.20	0.03	0.36	0.56	0.23
3	0.06	1.04	0.06	13.25	0.09	2.79
4	0.16	0.93	0.60	0.25	0.03	0.35
5	0.65	0.26	0.36	0.34	0.09	0.35
6	1.19	0.16	0.08	0.14	0.11	0.33
7	1.56	0.56	0.05	0.08	0.46	0.49
8	0.20	0.51	0.14	0.78	0.08	0.34
9	0.38	1.42	0.38	0.28	0.08	0.54
10	0.60	2.60	0.43	0.33	0.26	0.86
11	0.29	5.39	0.57	0.85	-	1.99
12	0.14	10.18	1.38	0.77	-	2.94
13	0.32	2.59	-	1.32	-	1.33
Overall MSE	0.40	1.69	0.35	1.40	0.24	0.91

yields the following expression:

$$MSE = \frac{1}{N_1 + N_2}\left[\sum_{j=1}^{N_1}\left(x_{1,j} - \bar{x}_1\right)^2 + \sum_{j=1}^{N_2}\left(x_{2,j} - \bar{x}_2\right)^2\right] = \frac{1}{N_1 + N_2}\left[N_1 \times MSE_1 + N_2 \times MSE_2\right]. \quad (15)$$

In practice, the recorded length of each test will not be the same. Also two of the five test subjects performed fewer tests, for example, $N_1 \neq N_2$. Therefore, the overall MSE was not an average of the individual MSEs.

It can also be seen from the tabulated results that the estimation accuracy degrades at speeds of higher than 10 km/h. This is likely due to the fact that at high speeds, accelerometers are functioning beyond their specification range of $\pm 2g$. However, it is only in an extremely rare case that the accelerometers are deployed beyond 10 km/h, which is not applicable for most patients.

5. CONCLUSION AND FUTURE WORK

This article presents *MobiSense*, a novel mobile body sensor network for ambulatory monitoring. This system resides in a mobile device and communicates with a set of sensor devices attached to the wearer's body. The proposed system has the ability to track wearers' activities displayed in the form of human animation on the user screen and is also equipped with resource management and reconfiguration schemes. The system itself has the potential to offer a wide range of benefits to patients, medical personnel, and society, through continuous monitoring in ambulatory settings, and early through detection of abnormal conditions of the patients.

We have described the system architecture, processes, functionalities, algorithms, and the prototype of the system, including its performance measurements. The performance indicators include power consumption of the sensor nodes, real-time and query latency of the system as well as the accuracy level

of our activity classifier scheme. The results indicate that our system is able to achieve an overall accuracy of 100% for classification of lying, sitting, standing, and walking activities. Furthermore, in our system the estimated walking speeds are used to distinguish between different types of movement activity (walking, jogging, and running), and the accuracy of its estimation has an overall mean-square error (MSE) of 0.91 $(km/h)^2$. We have also shown that the system is capable of preserving power and delivering raw sensor data to the application within 75 ms. Likewise, an experiment of critical-self-wake scheme as one of the most important features in the system, has been presented.

In future work, we shall incorporate additional sensors such as SPO_2 (Oxygen saturation) and BP (blood pressure) sensor into the system, and fuse the data from relevant sensors to determine the fitness level of a patient in real time at any given time. It is also of interest to implement a data logger in the mobile device, which functions as personal health profile of the wearer. Subsequently, we will look into transmitting sensor data to the central server and apply certain association mining techniques in order to find interesting patterns among patient, diseases, and activities involved, which may be useful for early diagnosis.

ACKNOWLEDGMENTS

The authors are grateful to the three anonymous reviewers for their constructive feedback, which considerably helped improve the article.

REFERENCES

ACTIGRAPH. 2009. http://www.mtiactigraph.com/ (last access 8/09).

BERKELEY. 2009. TinyOS community forum (online). http://www.tinyos.net. (last access 8/09).

BURRELL, J., BROOKE, T. AND BECKWITH, R. 2004. Vineyard computing: Sensor networks in agricultural production. *IEEE Pervasive Comput.*, *3*, 1, 38–45.

CARDIO MICRO SENSOR. 2009. http://www.cardiomems.com. (last access 8/09).

CROSSBOW TECHNOLOGY. 2009. http://www.xbow.com/. (last access 8/09).

DEVAUL, R. W., SUNG, M., GIPS, J., AND PENTLAND, S. 2003. MIThril 2003: Applications and architecture. In *Proceedings of the International Semantic Web Conference (SWC)*. 4–11.

DONG, L., WU, J. K., AND BAO, S. 2006. A Hybrid HMM/Kalman filter for tracking hip angle in gait cycle. *IEICE Trans. Inform. Syst.*, *E89-D*, 2319–2323.

EMBEDDED AND HYBRID SYSTEMS II (EHSII): A*STAR. 2009. http://www.ehs-sg.org/. (last access 8/09).

ESTRIN, D., GOVINDAN, R., HEIDEMANN, J., AND KUMAR, S. 1999. Next century challenges: scalable coordination in sensor networks. In *Proceedings of the 5th Ann. ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*. 263–270.

GAY, D., WELSH, M., LEVIS, P., BREWER, E., BEHREN, R. V., AND CULLER, D. 2003. The nesC language: A holistic approach to networked embedded systems. In *Proceedings of the ACM Sigplan Conference on Programming Language Design and Implementation (PLDI)*. 1–11.

GIANCOLI, D. C. 1998. *Physics 5th Ed*. Prentice Hall, Ch. 5.

JAMISON, D. T. 2006. Investing in health. In *Disease Control Priorities in Developing Countries*, 2nd Ed, D. T. Jamison, J. G. Breman, A. R. Measham, G. Alleyne, M. Claeson, D. B. Evans, P. Jha, A. Mills, and P. Musgrove, Eds. Oxford University Press, 1–34.

JOVANOV, E., LORDS, A., RASKOVIC, D., COX, P., ADHAMI, R., AND ANDRASIK, F. 2003. Stress monitoring using a distributed wireless intelligent sensor system. *IEEE Eng. Med. Engin. Biol. Mag.*, *22*, 3, 49–55.

LISZKA, K. J., MACKIN, M. A., LICHTER, M. J., YORK, D. W., PILLAI, D., AND ROSENBAUM, D. S. 2004. Keeping a beat on the heart. *Pervasive Comput. 3*, 4 , 42–49.

LO, B., THIEMJARUS, S., KING, R., AND YANG, G.-Z. 2005. Body sensor network—A wireless sensor platform for pervasive healthcare monitoring. In *Proceedings of the 3rd International Conference on Pervasive Computing*. 77–80.

LO, B. AND YANG, G. Z. 2005. Architecture for body sensor networks. In *IEE Proc. Perspec. Pervasive Comput.*, 23–28.

LORINCZ, K., MALAN, D. J., FULFORD-JONES, T. R. F., NAWOJ, A., CLAVEL, A., SHNAYDER, V., MAINLAND, G., MOULTON, S., AND WELSH, M. 2004. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Comput. 3*, 4, 16–23.

MALAN, D., FULFORD-JONES, T. R. F., WELSH, M., AND MOULTON, S. 2004. CodeBlue: An ad hoc sensor network infrastructure for emergency medical care. In *Proceedings of the Workshop on Applications of Mobile Embedded Systems (WAMES)*. 12–14.

MARSH, A. 2002. The E-care project—removing the wires. In *Proceedings of the International Conference on Computational Science (ICCS)*. 1012–1018.

MARTIN, T., JOVANOV, E., AND RASKOVIC, D. 2000. Issues in wearable computing for medical monitoring applications: A case study of a wearable ECG monitoring device. In *Proceedings of the International Symposium on Wearable Computers (ISWC)*. 43–50.

MATHIE, M. J. AND CELLER, B. G. 2001. A system for monitoring posture and physical activity using accelerometers. In *Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 3654–3657.

MED-ELECTRONICS INC. 2009. http://med-electronics.com/. (last accessed 8/09).

NEGI, R. AND RAJESWARAN, A. 2004. Capacity of power constrained ad hoc networks. In *Proceedings of IEEE INFOCOM*. *1*, 443–453.

PENTLAND, S. 2004. Healthwear: Medical technology becomes wearable, *IEEE Comput. 37*, 5, 34–41.

POLAR. 2009. http://www.polarusa.com/. (last accessed 8/09).

PRO MEDICUS LIMITED. 2009a. http://www.promedicus.com.au. (last accessed 8/09).

ROSS, P. E. 2004. Managing care through the air. *IEEE Spectr.* 14–19.

SILVA, B. D., NATARAJAN, A., MOTANI, M., AND CHUA, K.-C. 2008. Design considerations of body sensor networks. In *Proceedings of the Tenth IEEE International Conference on e-Health Networking, Applications & Services (IEEE Healthcom)*. 323–328.

SZEWCZYK, R., OSTERWEIL, E., POLASTRE, J., HAMILTON, M., MAINWARING, A., AND ESTRIN, D. 2004. Habitat monitoring with sensor networks. *Comm. ACM*, *47*, 6, 34–40.

TEXAS INSTRUMENTS. 2009. http://www.ortodoxism.ro/datasheets2/9/0oe5ltue2s77tejsk7j5xs0j72fy.pdf. (last accessed 8/09).

TEXAS INSTRUMENTS CHIPCON CC2420. 2009. http://focus.ti.com/docs/prod/folders/print/cc2420.html. (last accessed 8/09).

U.S. CENSUS BUREAU. 2009. http://www.census.gov/ipc/www/usinterimproj/. (last accessed 8/09).

WALUYO, A. B., PEK, I., YING, S. JIANKANG, W., CHEN, X., AND YEOH, W.-S. 2008. LiteMWBAN: A lightweight middleware for wireless body area networks. In *Proceedings of the 5th International Workshop on Wearable and Implantable Body Sensor Networks (BSN'08)*. 141–144.

WANG, C. C., CHIANG, C. Y., HUANG, C. N., AND CHAN, C. T. 2008. Development of a fall detecting system for the elderly residents. In *Proceedings of the 2nd IEEE International Conference of Bioinformatics and Biomedical Engineering*. 1359–1362.

WELCH, J., GUILAK, F., AND BAKER, S. D. 2004. A wireless ECG smart sensor for broad application in life threatening event detection. In *Proceedings of the 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 3447–3449.

YANG, G.-Z., LO, B., WANG, J., RANS, M., THIEMJARUS, S., NG, J., GARNER, P., BROWN, S., MAJEED, B., AND NEID, I. 2004. From sensor networks to behaviour profiling: A homecare perspective of intelligent building. In *Proceedings of the IEE Seminar for Intelligent Buildings.*

YEOH, W. S., WU, J. K., PEK, I., YONG, Y. H., CHEN, X., AND WALUYO, A. B. 2008. Real-time tracking of flexion angle by using wearable accelerometer sensors. In *Proceedings of the 5th International Workshop on Wearable and Implantable Body Sensor Networks (BSN'08)*. 125–128.

ZHOU, H. Y. AND HU, H. S. 2004. A survey—human movement tracking and stroke rehabilitation. Tech. rep., CSM-420 ISSN 1744–8050.

Received September 2008; revised March 2009; accepted August 2009