A Rule-based Acceleration Data Processing Engine for Small Sensor Node

Kenji KODAMA
Graduate School of Engineering, Kobe University
kodama@stu.kobe-u.ac.jp

Naotaka FUJITA
Graduate School of Engineering, Kobe University
nfujita@stu.kobe-u.ac.jp

Yutaka YANAGISAWA
NTT Communication Science Laboratories
yutaka@cslab.kecl.ntt.co.jp

Tsutomu TERADA
Graduate School of Engineering, Kobe University
tsutomu@eedept.kobe-u.ac.jp

Masahiko TSUKAMOTO
Graduate School of Engineering, Kobe University
tuka@kobe-u.ac.jp

ABSTRACT

In recent years, various small sensor nodes have been developed to recognize situations and events occurring in the real world for the development of context-aware systems. We consider that the acceleration sensor is one crucial element for recognizing various types of situations because it has rich and simple information. An application system using acceleration data has the following requirements on sensor node: 1) rapid processing of data without large memory since the amount of acceleration data is much greater than the amount of other sensor data; 2) a node that can reduce the amount of data sent to a server; and 3) systems that can be easily configured by users with low cost. Existing sensor nodes, however, do not have enough functions to satisfy these requirements. In this paper, we propose a rule-based data processing engine for processing acceleration data. Our proposed system can rewrite the rules on each node with a few bytes of data. To evaluate our proposed mechanism, we experimented with our rule-based system implemented on our small sensor node called MoCoMi-Chip.

General Terms

Design, Implementation, Experimentation, Performance

Keywords

Acceleration Sensor, Rule Base, Sensor Node

1. INTRODUCTION

Recent technological advances have created pervasive computing systems using many small computer nodes with various types of sensors to obtain events that occurred in the real world. The nodes enable us to develop new services, based on huge amounts of information such as contexts, events, and situations.

Many existing nodes have several types of sensors such as temperature, proximity, acceleration, light, pressure, and magnetic to extract beneficial real information [1]. Among the above, the acceleration sensor is one of the most significant examples for obtaining such data as dynamic motions, both of humans and objects, in real time. Most existing sensor nodes have acceleration sensors.

Generally, the amount of acceleration data obtained in a second is much larger than the other sensor data because an acceleration sensor obtains data in high frequency. The increase in the amount of data causes a corresponding raise in communication cost between sensor nodes. Another problem concerns users: they must frequently adjust each setting of the sensor nodes to accurately process the acceleration data because acceleration data are strongly influenced by the real world. For this reason, acceleration sensor nodes require operation-specified acceleration data.

We consider four requirements: 1) simple description of program; 2) reconfigurable program; 3) promptitude; and 4) low communication cost. Existing sensors, however, use the same method to process all types of sensors, even though acceleration sensors have quite different features from other sensor nodes. In other words, previous sensor nodes have functions that satisfy the requirements of acceleration sensors because they use acceleration data as well as other sensor data sampled in low frequency. To deal with the acceleration data on the sensor nodes, the data processing system on a node must introduce specified mechanisms to process the acceleration data on each sensor node.

In this paper, therefore, we propose a rule-based data processing system specified to process the acceleration data on small sensor nodes. Using rule description, users describes the operations of sensor nodes simply and compactly. Furthermore, to modify the rules, users can change operations dynamically in real time. Our proposed system can operate sensor data in high frequencies because the engine uses a

simple processing method. The engine can reduce the communication cost because sensor nodes can only send sensor data or contexts when the engine detects events. Moreover, we describe an implementation of our rule-based system on our small sensor node called a Motion sensing and Communication Minimized Chip (MoCoMi-Chip). We also experimentally evaluated the performance of our proposed mechanism based on the above four system requirements.

This paper is organized as follows: we survey related works in Section 2. In Section 3, we introduce our approach and describe the design of our proposed system in Section 4. In Section 5, we show an evaluation of our system. Finally, we conclude in Section 6 with our summary and directions for future work.

2. RELATED WORKS

In this section we show related systems using sensor nodes. First, we introduce systems that obtain context in the real world. Second, we introduce database systems that process sensor data on sensor networks. Finally, we introduce rule-based systems for sensor nodes.

Many researchers have attempted to obtain the contexts of humans or objects using sensor nodes. In these systems, acceleration data are crucial to obtain both the contexts and the situations. For example, smart object service supports human daily life. MediaCup attached sensor nodes to cups to obtain contexts [2]. DigiClip uses sensor nodes attached to papers to manage them [3]. User activity recognition systems also use acceleration data for analysis. [4] proposed methods to estimate the motion of a person using attached sensor nodes. These systems process the data stored in server computers without processing the sensor nodes.

To store sensor data in a server, each node must send data with a wireless network. As the amount of sensor data obtained by sensor nodes increases, the amount of data sent to the server also increases. TinyDB can reduce the amount of sensor data using acquisitional query processing (ACQP) in sensor networks [5]. TinyDB can process a query described in a SQL-like query language. The user can indicate the minimum set of necessary sensor data to send to the server. However, TinyDB does not suit acceleration data to estimate the motion context due to their features.

In the rest of this section, we mention the reconfigurable mechanisms for sensor nodes. Over-the-air programming (OTAP) is proposed to rewrite programs stored on many sensor nodes in wireless networks [6]. For example, both MICA MOTE and Smart-Its Particle introduced OTAP to reduce rewriting cost [7, 8]. However, since OTAP spends much communication cost and cannot rewrite the program in real time, this method cannot be adapted to sensor nodes, which require quick replacing.

Some pervasive services using rule-based systems have been proposed to adapt various environments for sensor nodes. A rule-based system can operate small devices as an event-driven system and can change programs quickly. AhroD and DYCOM use rule-based systems to operate small sensor nodes for pervasive services [9, 10]. AhroD is a ubiquitous

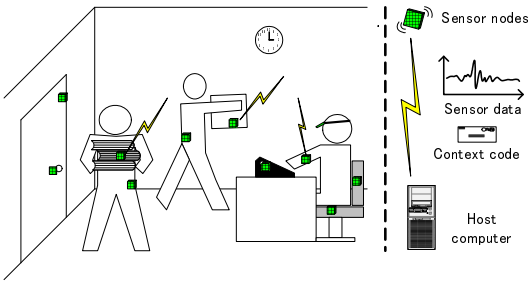

Figure 1: An application image.

computing device using event-condition-action (ECA) rules whose description simply and compactly describes device operation. AhroD has operation rules that concern which application should be executed and processes them in sequence. However, AhroD cannot process sensor data because it uses binary signals to evaluate input signals. DYCOME is a context-oriented switcher using a rule-based system to process sensor data for sensor nodes. Processing raw sensor data, DYCOME can dynamically switch an active application at an event. However, the rule description in DYCOM cannot describe a motion event because it only processes raw acceleration data and other sensor data.

3. APPROACHES

As mentioned in Section 1, we propose a method to operate small autonomous sensor nodes that can process acceleration data. In this section, after describing several examples of our supposed application systems and the framework of a sensor network, we discuss the system requirements to process acceleration data on our sensor nodes for application systems. Finally, we show an outline of the method to satisfy the requirements.

3.1 Example Application Systems

We focus on the sensor nodes attached to various indoor objects. The application systems use collected data from each sensor node to extract events and context from the real world.

Figure 1 shows a focused example of application systems. Here, each person has a sensor node, and some objects also have a sensor node in the office. Each sensor node obtains various types of sensor data, and the node extracts beneficial data from the obtained sensor data. After extraction, the node sends the data by a wireless network to a host computer. The host computer integrates the received sensor data to calculate complex moving situations or events that each node can extract. The host computer also sends the information to application systems. As a result, an application system can provide rich context-aware services.

Generally, each node has many types of sensors, such as acceleration, temperature, light, and so on. Acceleration sensors are crucial to obtain data to recognize real-world situations since they can catch the motions or tilts of objects in detail. Most context-aware systems adapt acceleration sensors to extract contexts such as human activities, situations, and events. The acceleration sensor nodes in these systems must have higher data processing capability than other sensor nodes.

On the other hand, attachable sensor nodes have poor resources of calculation and battery because they must be small and low cost. In other words, acceleration sensor nodes must process sophisticatedly with small resources. To satisfy these requirements, we must develop a processing method to reduce processing cost on a node.

3.2 System Requirements

Here we discuss the requirements of the method to process acceleration data on a small sensor node.

3.2.1 Requirements from Application Systems

- Simple description
  In the supposed application, the system is used by many people, some of whom do not have experience configuring sensor nodes. Therefore, sensor node programs should be thumbnail descriptions.
- Reconfigurable
  In the example application systems, sensor node programs may be different from each node due to provided services or attached places. The application systems should change the method to process the sensor data on each node for each purpose. For example, if a user wants to add a new event to detect, the program to process the data on the sensor nodes must be rewritten in the routine. The features of the sensor data are different from each attached node. Moreover, each node has different hardware features so that we must give individual settings for each node. Since sensor nodes often need reconfiguration, they require great reconfiguration cost. To adapt the processing method to frequency changing situations, it is necessary to introduce a mechanism to rewrite the program on a node without much communication and processing cost. To reconfigure the sensor nodes easily by wireless, communication costs must be low.

3.2.2 Requirements from Acceleration Data

- Promptitude
  The acceleration data have different features from the other types of sensor data, for example, high frequency sampling, fluidity of data, and independence of data. In these ways, acceleration sensor nodes need high frequency sensing and rapid response. For example, to track such human motions as walking and running, the nodes must sense the acceleration data of the person's body in 10-100 Hz. On the other hand, temperature and light are sampled up to 10 Hz. If an alert system detects such anomalous events as collisions or the fall of products in a factory, the system must response rapidly until a person helps.
- Low communication cost
  In general, the amount of acceleration data is larger than other sensing data because the acceleration sensor must obtain data to trace a motion in high frequency. If a node sends all obtained data to the host computer by a wireless network, the amount of data sent from a number of nodes may exceed the capacity of the wireless network because a small sensor node has only a low band rate wireless communication device. To avoid such situations, acceleration sensor nodes must reduce their transmitted data; that is, sensor nodes must only send necessary data. Thus, sensor nodes should process the acceleration data on the sensor nodes and minimize the transmitted data. However, processing must be simple, because the resources of the sensor nodes are small.

3.3 Our Approach

In this paper, we propose an acceleration data processing system that can satisfy the above four requirements for small acceleration sensor nodes. Here, we briefly show how to satisfy the requirements with our system. For a simple description, we adapt rule-based language to process the sensor data. Our system uses an If-Then rule that consists of conditions and actions. The condition of the rules is based on acceleration data. If a condition is implemented, a corresponding action is executed. Using rule-based language, we can rewrite the sensor node program. To add or delete rules, application systems can quickly change the operation of sensor nodes and reduce the communication cost of reconfiguration. For promptitude, we adapt threshold comparisons to decide the conditions, which are described as feature values: instantaneous value, average value, sum of deviation, derivative value, and number of counters. Operating on event-driven to process the rules based on acceleration data, sensor nodes can reduce communication costs. Most sensor nodes only send a context as a short piece of text data to a host computer but do not send raw long sensor data.

4. RULE ENGINE FOR ACCELERATION DATA

4.1 Rule Engine Design

Since both sensing acceleration and processing data are elemental tasks on sensor nodes, the engine must have their tasks before every evaluation process cycle. To operate sensor nodes at accurate intervals, the engine uses a hardware timer for sensing in a cycle. In other words, since the hardware timer announces when the next cycle starts after processing all rules, the engine can periodically operate the processes on a sensor node. The engine processes the stored rules in order. Having several rules, the sensor nodes can operate for several situations. The engine allows combinations of conditions or actions for rule flexibility.

4.2 The flow

Figure 2 shows the essential processing flow of our designed rule engine on a sensor node.

1. The engine obtains a piece of acceleration data from the sensor device on the node, before calculating the feature values of the acceleration data as preparation. The feature values are used to recognize the motions, tilts, and states of the sensor node.
2. The engine fetches a rule one by one, before comparing the condition with extracted feature values in the previous step. If the engine finds a matched condition in a rule description with a feature values, the engine checks a next rule. In another case, if the checked condition is a combined condition, the engine fetches each rule from the combined condition and evaluates every fetched condition recursively.
3. If all fetched conditions are matched with the values, the engine does the action. The engine also checks the next from the first described rule to the end of the rules one by one.

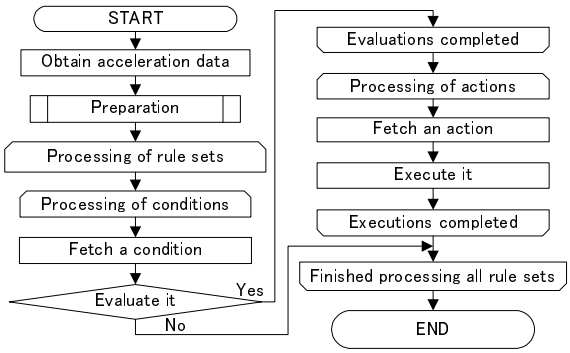

Figure 2: System flowchart.

4. The engine waits until an interruption from the hardware timer when the engine has completely evaluated the rules.

As described above, the engine processes the rules based on acceleration data in a node.

4.3 Condition

Traditional systems capture motions or situations from acceleration data to calculate average, variance, deviation, FFT, and Support Vector Machine (SVM) as feature values. However, sensor nodes cannot calculate such sophisticated processes as FFT and SVM in real time.

Our systems calculate next three feature values of the acceleration data for high-speed evaluation of the conditions at preparation. The engine can simply calculate these feature values on the sensor nodes. The average can show tilts or smooth sensor data. Variance is an available value to detect the presence of motion, but it cannot be calculated rapidly using a low cost MCU. The conditions of our rules use a sum of deviation that resembles variance to detect the presence of motion. The engine calculates an average and a sum of deviation in the last 10 samples. To detect rapid motions such as object hits, the conditions also use a derivative value. In addition, the engine uses four counter values (condition counters) to describe continuous situations further to the feature values for evaluation of the conditions. The engine can increment and reset the condition counters by executing an action. Using the condition counters, we can describe the condition of continuous situations: if a state occurs more than once, the engine measures the number. The engine allows descriptions of conditions that regard a continuous state as an event. Using this regard, we can describe the condition of an event start where a state is detected continuously. The engine evaluates conditions to compare threshold and feature values by simple processing.

4.4 Action

The engine allows the execution of next ten actions by processing the rules for sensor nodes.

Sending acceleration data is one basic operation of sensor nodes. This action sends raw acceleration data, feature values, and counter values. The engine must also send acceleration data continuously when the host computer asks for stream data. To reduce the communication cost, the engine allows the contexts to be sent that detect events and

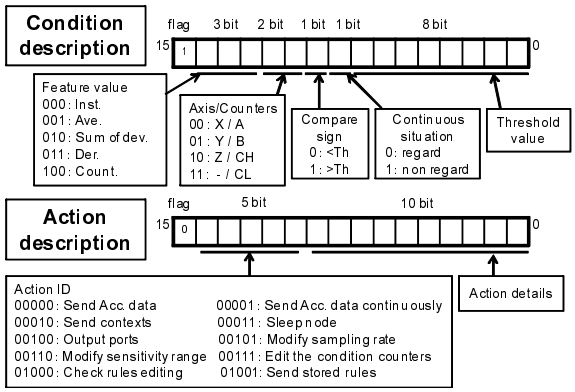

Figure 3: Format of rule description.

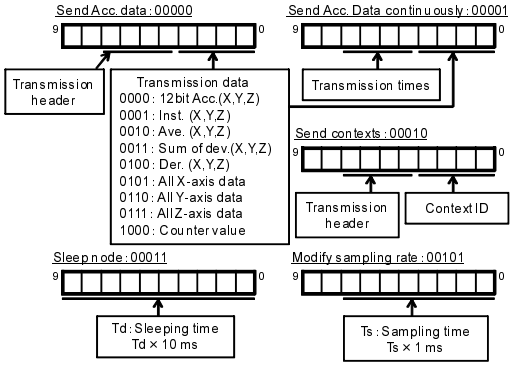

Figure 4: Part of detail formats of action description.

the situations of sensor nodes. Putting the node to sleep is important to conserve the battery. In addition, the engine allows output ports to indicate LEDs, modifying the sampling rate and sensibility range to adapt to situations. The condition counters are edited by action to increment and reset for continuous situations. The engine also allows checking for the presence of rules editing and sending stored rules on sensor nodes.

4.5 Rule Editing Mechanism

The engine runs the rule-editing mechanism to follow the nxet four steps. First, a sensor node checks the presence of rules editing to send a request to a host computer as an action or a routine. Second, after receiving the request, the host computer sends the presence and areas of rules editing. Third, if the rules need to be edited, the sensor node sends a request of rules forwarding. Finally, after rules forwarding, the sensor node verifies the rules by check sum and sends a finished message or a retransmission request.

4.6 Rule Description

The rules consist of combinations of conditions and actions. We use a binary rule that can easily evaluate the conditions to compare the threshold value with feature values. The rules are described based on the formats of the rule description shown by Figures 3 and 4. The descriptions of the conditions and actions are fixed-length two-byte data. Users can program multi-condition and multi-action using a MSB flag. In fact, continuing conditions that have MSB flags are evaluated as “AND condition.” To program an “OR condition,” a user should describe each set of rules. Continuing actions are executed for each action.

A description of a condition consists of five components: kind of values, axis or kind of counter, sign, and regard of continuous situation and threshold values. The threshold value is the 8-bit abbreviated feature values or a value of the condition counters. The feature values are calculated at the preparation based on the acceleration data.

A description of the condition consists of an action ID and a configuration of an action detail. Except for MSB, high 6 bits are allocated for each action, and the low 10 bits show the configuration of an action detail. Figure 4 shows part of the detail formats of the action description. The configuration of an action detail differs by each action. For example, for an action that sends acceleration data, users can select sending data to each axis and each kind of data.

5. EVALUATION

In this section, we experimentally evaluate the performance of our proposed mechanism from the viewpoint of the four system requirements. In the experiments, we attached the MoCoMi-Chip to humans and objects. Table 1 shows specifications of our developed chip, and Table 2 shows the evaluation results. The engine tried to detect occurred events based on rule descriptions. Figure 5 shows our chip attached to a pen, and Figure 6 shows acceleration data obtained from the chip while writing. The engine had rules that are provided accurate descriptions based on exploratory experiments. The amount of a transmission packet between node and server was set to 15 bytes, and the payload was set to 8 bytes.

5.1 Simple Description

To evaluate the simplification of operation description, we compared the amount of code written with our description language with the amount of code description in C language. Because C language is the most popular language for programming MCU, we adapt a language to compare the simplifications. Using the rule description, a rule is composed of conditions and actions described as binary code whose size is only two bytes. A set of rules is described as four bytes in hex format. When we can describe an operation to detect an event, the amount of the rule description is 20 % of the amount of the code described in C language. Note that C language in available to describe the details of operation using calculations, control statements, and functions for sensor nodes. For sensor node operations that only have cyclical routines, however, our rule description is simpler than C language for users.

5.2 Reconfigurable

We measured the time of the rule editing routine to test a real-time reconfiguration. In this experiment, our chip had 32 rules and edited eight and 16 rules into new rules. From this experiment result, the routine takes 12.4 ms when changed eight rules, and 14.5 ms when changed 16 rules. This result suggests that the engine can edit the rules in the sampling cycle required to sense the motions of humans and objects. In the other words, the engine can change the rules dynamically without pausing operation for sated the sensor nodes in the supposed applications.

5.3 Promptitude

To evaluate the promptitude, we measured the operation time of our proposed system. The operation times included

Table 1: Specifications of MoCoMi-Chip.

MCU	8051 compatible microcontroller
Operation frequency	16 MHz
Program memory	4 KB
Data memory	256 byte
Radio transceiver	nRF2401
RF data rate	1 Mbps/250 Kbps
Communication distance	approx. 25 m
Technical regulations conformity certification	2.4 GHz band wide-band low-power data communication system in Japan
Interfaces	UART, SPI, Digital I/O, PWM
Acceleration full scale	±2 G/6 G selectable
Acceleration resolution	12 bit
Consumption current	Transmission: Typ.20 mA, Sleep: 4 μA
Power supply	DC 3 to 9 V
Overall size	20×20×3.9 mm
Weight	approx. 1 g

Table 2: Evaluation results

Name of detected events	Tx data [Byte]	Amount of data rate	Number of occurred events	Processing time [ms]	Number of rules	Amount of code In C language [Byte]
Transmit raw 3-axis acceleration data	75000	100	-	-	-	-
Start of free fall	150	0.2	10	2.47	7	138
Fall to floor	585	0.78	39	2.49	12	294
Spots on a dice	270	0.36	18	2.58	18	396
Opening and closing a door	180	0.24	12	2.33	6	130
Opening and closing a drawer	180	0.24	12	2.47	10	216
Start of writing	1575	2.1	105	2.30	4	84
Standing and sitting of human	435	0.58	29	2.38	8	169
Human moving	2010	2.68	134	2.11	3	59
Stop hand motions	240	0.32	16	2.30	4	88

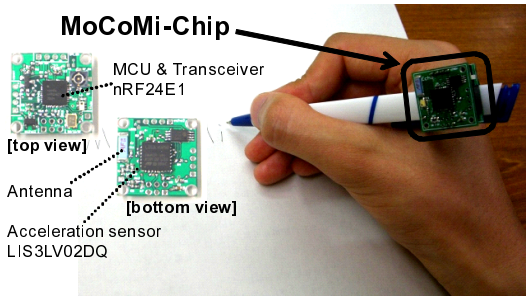

Figure 5: MoCoMi-Chip mounted on a pen

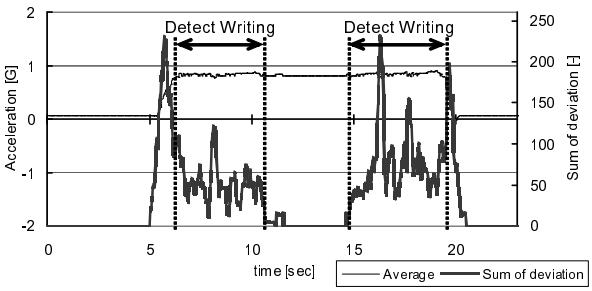

Figure 6: Acceleration data in pen operation

both obtaining the sensor data and sending messages of events as a context to the server. To measure operation time, we used a hardware timer on MCU in our chip. In this experiment, the engine detects such events as opening and closing a door, starting a free fall of a chip, writing, and a moving human. Table 2 shows the processing time results. In most cases, the processing time was less than 2.5 ms. In other words, the engine can process the operation to detect events with over 400 Hz. To obtain human motions, an event detection system must generally process the detection operation over 10-100 Hz. Thus, the results indicate that the engine has enough power to detect human events in real time. In addition, we measured the processing time of each part of the operation. The longest processing time was the transmitting operation, which required 1.0 ms. Evaluating the rule's condition required about 0.04 ms. To completely finish processing at a cycle, we must strictly describe the rule and reduce the number of occurred events found within a process cycle. This technique enables us to reduce the processing time although we increase the number of rules described in a sensor node.

5.4 Low Communication Cost

To evaluate the communication cost, we experimentally measured the amount of data sent from a sensor node to the server computer. In this experiment, we compared the case of event contexts as a text with event contexts and raw three-axes acceleration data. We measured the amount of transmission data by transmitting event contexts and raw three-axes acceleration data and compared them to evaluate communication cost. For the settings, our chip has rules to detect the start of each event; an event occurs 10 times per 100 seconds. Table 2 shows the results of this experiment. The amount of transmission data was less than 2,000 bytes when our proposed mechanism transmitted the context of an event. On the other hand, for transmitting raw three-axes acceleration data within 100 seconds, the amount of transmission data was 75,000 bytes. This result indicates that our mechanism can reduce communication cost by one hundredth of the transmission data. Therefore, we consider that the mechanism can save more power of the sensor nodes than the naive transmission method.

5.5 Discussion

We detected the events of humans and object motions using our proposed system. However, it could not detect complex situations such as human activities because the engine uses feature values calculated by simple processing. To detect complex situations, we must configure the rules to transmit acceleration data when a motion occurs and calculate them on a host computer, as in traditional systems. Therefore, an operation that combines our proposed systems for sensor nodes and host processing operation is available to conserve the batteries of sensor nodes and communication costs.

We implemented the engine to our chip on an Intel 8051 compatible MCU. The program size of the engine was 4 KB. Therefore, we can implement it on most conventional sensor sensor nodes because they have larger resources than our chip.

6. CONCLUSION

In this paper, we proposed and designed a rule engine to process acceleration data on small sensor nodes. We developed an acceleration sensor node called MoCoMi-Chip and evaluated the performance of the engine with a node from the viewpoint of the four system requirements. Evaluation results show that the engine satisfies the four requirements. In the future, we plan to develop a system that easily describes and generates the rules. We also plan to develop a rule engine that processes other sensor data.

7. REFERENCES

[1] Beigl, M., Krohn, A., Zimmer, T. and Decker, C.: Typical Sensors needed in Ubiquitous and Pervasive Computing, In Proc. of INSS 2004, pp. 153–158 (2004).

[2] Beigl, M., Gellersen, H.-W. and Schmidt, A.: Mediacups: Experience with Design and Use of Computer-Augmented Everyday Artefacts, Computer Networks, vol.35, No.4, pp.401–409 (2001).

[3] Decker, C., Beigl, M., Eames, A. and Kubach, U.: DigiClip: Activating Physical Documents, In Proc. of the Intl. Conf. on Distributed Computing Systems Workshops (ICDCSW'04), pp.388–393 (2004).

[4] Kawahara, Y., Kurasawa, H. and Morikawa, H.: Recognizing User Context Using Mobile Handsets with Acceleration Sensor, In Proc. of the IEEE Intl. Conf. on Portable Information Devices (IEEE Portable 2007) (2007).

[5] Madden, S., Franklin, M., Hellerstein, J. and Hong, W.: TinyDB: an Acquisitional Query Processing System for Sensor Networks, ACM Tran. on Database Systems, Vol.30, No.1, pp.122–173 (2005).

[6] Kulkarni, S., Wang, L.: MNP: Multihop Network Reprogramming Service for Sensor Networks, In Proc. of the IEEE Intl. Conf. on Distributed Computing Systems, pp. 7–16 (2005).

[7] Crossbow Technology, inc. http://www.xbow.com/.

[8] Holmquist, L., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M. and Gellersen, H.-W.: Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts, In Proc. of the Intl. Conf. on Ubiquitous Computing (UbiComp 2001), pp. 116–122 (2001).

[9] Terada, T., Tsukamoto, M., Hayakawa, K., Yoshihisa, T., Kishino, Y., Nishio, S. and Kashitani, A.: Ubiquitous Chip: a Rule-based I/O Control Device for Ubiquitous Computing, In Proc. of the Intl. Conf. on Pervasive Computing (Pervasive 2004), pp. 238–253 (2004).

[10] Koizumi, K., Sakakibara, H., Iwai, M. and Tokuda, H.: A Context-Oriented Application Switching Mechanism for Daily Life Supports, the Intl. Conf. on Ubiquitous Computing (UbiComp 2005), Poster Session (2005).