

Design of a Wearable Sensor Badge for Smart Kindergarten

Sung Park, Ivo Locher, Andreas Savvides,
Mani B. Srivastava

*Networked Embedded Systems Laboratory
Electrical Engineering Department, UCLA
{spark, ilocher, asavvide, mbs}@ee.ucla.edu*

Alvin Chen, Richard Muntz, Spencer Yuen

*Multimedia Systems Laboratory
Computer Science Department, UCLA
{alchemy, muntz, spencery}@cs.ucla.edu*

Abstract

We describe the system design of “Smart Kindergarten”, a smart problem solving environment for early childhood education. Two crucial building blocks that make up Smart Kindergarten are iBadge, a lightweight power-aware sensor-instrumented badge, and Sylph, a sensor middleware infrastructure. Equipped with sensors, radio, and DSP, iBadge has been designed to be worn by kids and teachers to unobtrusively capture the interaction between the kids and their surroundings. The captured data is then wirelessly transmitted to Sylph, a middleware infrastructure, which provides means of accessing the data in coherent and user-friendly manner. In this paper, we present the design challenges of iBadge and Sylph and our efforts in integrating these two systems.

1. Introduction

Much advancements in low-power embedded processors, radios, and micro-mechanical systems (MEMs) has enabled the development of tiny wireless embedded computing device with integrated sensors [1]. With their focus on applications requiring tight coupling with the physical world, these wireless embedded computing device pose significantly different design, implementation, and deployment challenges. In the ongoing “Smart Kindergarten (SmartKG)” project [2] at UCLA, we are exploring new applications for this emerging wireless embedded sensor technology. Our vision in Smart Kindergarten is to utilize the wearable sensing and computing architecture with the supporting background infrastructures for exploring new paradigm in wireless communication of “person to physical world” interaction, as opposed to the conventional

This material is based upon work supported by the National Science Foundation under Grant No. ANI-0085773, IIS-0086116, EAR-9817773. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF)

“person to computer” and “person to person” interactions. Smart environments instrumented with sensor-and-wireless-enhanced objects would be able to sense events and conditions about people and objects in the environment.

Two crucial building block of this environment are iBadge, a light-weight power-aware sensor-instrumented badge that is to be worn by kids and teachers and Sylph, a sensor middleware infrastructure. In SmartKG, through the use of iBadge along with other sensors, we intend to unobtrusively capture the interaction between the kids, the teachers, and the common classroom objects. The captured data is then wirelessly communicated to the wired middleware infrastructure, which makes up Sylph. Sylph is responsible for fusing the captured data into coherent information, which can readily assist the teachers in evaluating and assessing the progress of children’s learning process and their social behavior. In this paper we present the system design of the iBadge and how it interacts with the Sylph middleware infrastructure. The success of the project relies on tight integration of the two distinct systems, the wearable hardware architecture and the background middleware infrastructure. Compared to our previous work [3] where we have emphasized the middleware infrastructure of SmartKG, our focus in this paper is on the wearable hardware architecture of iBadge. In the following sections we present the design, the motivation, the challenges, and our methodology for the implementation of iBadge and Sylph.

2. System architecture of Smart Kindergarten

In SmartKG, we target the early childhood education environment as a testbed, where we try to provide parents and teachers with the abilities to comprehensively investigate students’ learning processes. The kind of questions that we hope to answer ranges from evaluations of students’ progress such as “How well is student A spelling a word?” and “How much time does student B spend on solving a math problem?” to evaluations of students’ social behaviors such as “Do students tend to be in a

group with the same ethnic background?” and “Is student C usually isolated?”

The infrastructure of SmartKG has been designed to collect, manage, and fuse the information of the sensors to interpret and present the information in a logical and user-friendly manner. In this section we discuss the infrastructure that constitutes SmartKG and how iBadge fits into the overall picture. Figure 1 shows the two major components of the SmartKG architecture, Sensing Infrastructure, and Middleware Infrastructure.

- **Sensing infrastructure:** A suite of sensors that comprise the Sensing Infrastructure includes video cameras, microphones, motion detectors and iBadges. iBadges are the most critical elements that make up the Sensing Infrastructure. Either worn by the students or embedded in class objects such as toys or books, iBadges will enable higher layer services (middleware, applications) to discover and keep track of context information about the students and the toys.
- **Middleware infrastructure “Sylph”:** Sylph provides various services that process, store, fuse, manage, and present the data that either have been collected or are being collected from Sensing Infrastructure. On one hand, Sylph manages the interaction between applications and other higher-level services such as speech recognition, sensor data storage and sensor data browsing. On the other hand, Sylph is responsible for managing sensor devices by loading the interface to sensor objects called “sensor modules”.

3. iBadge design overview

3.1 iBadge architecture

iBadge has been designed to support various sensing services to higher layers while meeting power and size

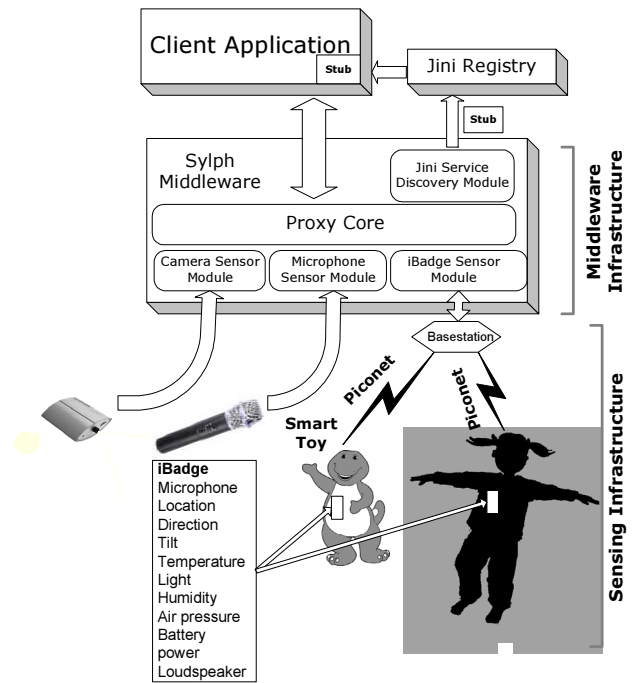


Figure 1. Smart Kindergarten system architecture

constraints. Some of the most critical design constraints of iBadge include form factor, energy consumption, cost, design flexibility as well as various sensing functionalities. iBadge must be light and small enough for children to wear. Finding the optimal location of iBadge on a kid’s body is still subject to investigation. In a typical operational scenario for Smart Kindergarten, iBadge requires to operate for 4 to 6 hours a day. Although a logical design solution to satisfy power and form factor constraints leads to an ASIC design, an ASIC solution does not fully satisfy the cost and the flexibility constraint. Instead, iBadge was designed as a board level integrated system consisting of

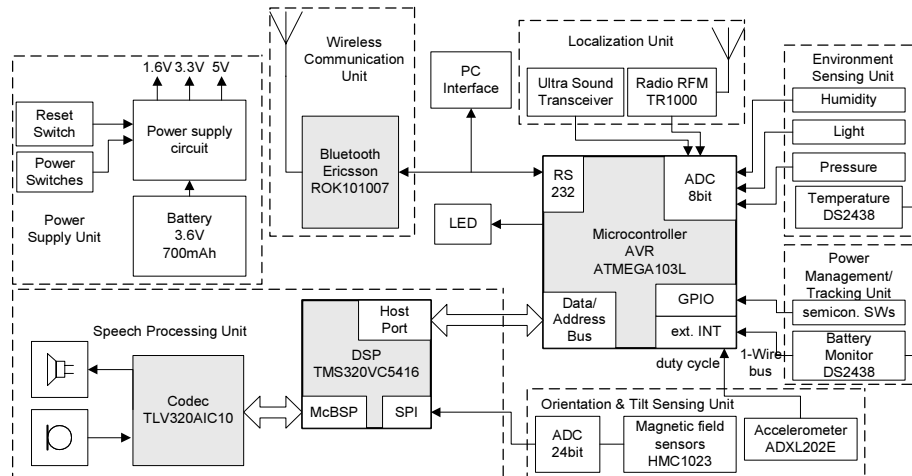


Figure 2. iBadge architecture

low-power standard chips (DSPs, microcontrollers) with off-the-shelf components. The resulting design allowed substantial cost saving as well as the flexibility to change components without considerable redesign efforts. However, iBadge's power consumption was a factor that required much attention resulting in addition of a rigorous power management and tracking unit to the overall design. The power tracking unit supplies knowledge of overall power consumption of the circuit as well as the power consumption of the functional units.

Another difficult aspect of iBadge design is that, at current stage, the kinds of information and data required by higher layers is still unclear. To our knowledge, the level of information details that we are planning to gather and analyze has never been tried before. iBadge will be one of the first experimental platform that we will be able to study and analyze for the usefulness of different functionalities and sensor services. iBadge's design reflects this experimental aspect through the integration of various sensing functional units along with the power management/tracking functional unit. Shown in Figure 2 is the block diagram of the iBadge architecture with its functional units. The heart of the iBadge is the Atmel ATMEGA microcontroller. It controls various sensors that makes up the Environment Sensing Unit, Localization Unit, and Orientation/Tilt Sensing Unit, and interfaces the Bluetooth module, which constitutes the Wireless Communication Unit. While ATMEGA is responsible for managing the interface of various functional units, Speech Processing Unit deals with algorithms that require heavy computation such as speech compression and orientation calculation. The Bluetooth module, which serves as the wireless data channel to the background infrastructure, is a programmable stand-alone unit with a full implementation of Bluetooth stack. Important details of the different functional

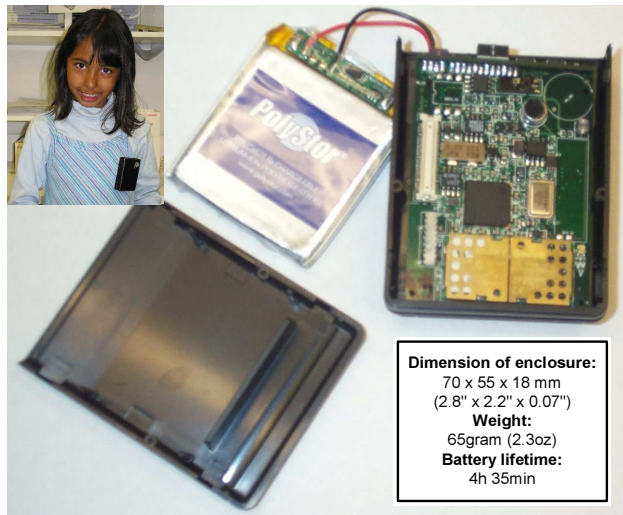


Figure 3. iBadge enclosure

units will be discussed in section 0.

Figure 3 shows a photograph of iBadge with its enclosure and battery. All three components together weigh 65 grams (2.3 ounces), and the average battery lifetime is 4 hours and 35 minutes.

3.2 Communication/Computation tradeoff

A key to the design of iBadge Software Interface (section 5.3) is partitioning of the functionalities across the boundary between iBadge and Sylph. Many of the sensor services can both reside in the Sylph middleware and iBadge Software Interface. These services include speech recognition service where front-end processing can be performed either at iBadge or in Sylph (section 4.2). The same is true for orientation/tilt sensing where the sensor information from acceleration and magnetic sensors can be combined at either places, or localization where the location calculation can be performed at both places. In order to decide which services should be performed at what level, the power consumption ratios between communication and computation must be considered. The following equations highlight the communication/computation tradeoff by roughly estimating the energy ratio of transmitting one bit across the wireless link to executing an instruction either at DSP or ATMEGA.

$$E_{BT} = \frac{P_{TX}}{DataRate} = \frac{26mA \cdot 3.3V}{64kbit/sec} = \frac{0.0858J}{64kbit} = 1.3 \times 10^{-5} J/bit \quad (1)$$

$$E_{DSP} = \frac{P_{DSP}}{ClockRate/CPI_{avg}} = \frac{99.8mA \cdot 3.3V}{80MHz} = \frac{0.329J}{2cycles/instr} = 8.2 \times 10^{-9} Joules/instr \quad (2)$$

$$E_{ATMEGA} = \frac{P_{ATMEGA}}{ClockRate/CPI_{avg}} = \frac{47.8mA \cdot 3.3V}{4MHz} = \frac{0.157J}{1cycles/instr} = 3.9 \times 10^{-8} Joules/instr \quad (3)$$

$$E_{DSP} : E_{ATMEGA} : E_{BT} = 1 : 4.75 : 1585 \quad (4)$$

E_{BT} is the energy needed to transmit a single bit using Bluetooth radio with HV1 packet format through a single SCO channel (64kbps; www.bluetooth.org). E_{ATMEGA} and E_{DSP} represent the energy needed to execute a single instruction with ATMEGA and DSP respectively. The ratio shown in equation 4 indicates that the energy cost of transmitting a single bit is in the order of three magnitudes more expensive than executing a single instruction. Generally speaking if the functions implemented in ATMEGA or DSP can reduce the number of bits required to transmit with the ratio of one bit to 300 instructions for ATMEGA and 1500 instructions for DSP, implementing the functions in iBadge will result in energy saving. Power measurements from PMTU (section 0) will be a good

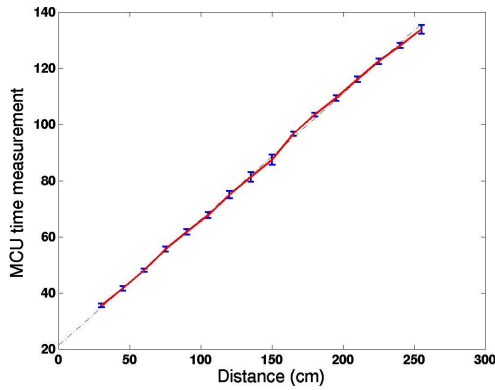


Figure 4. Distance measurement

basis for studying the tradeoffs between communication and computation.

4. iBadge main functional units

As shown in Figure 2, iBadge is the integration of different functional units. In the following subsection, we describe the functionality of the major units in more detail. One of the most important features of iBadge is its ability to provide precise location and orientation information of the wearer. This information is then used to identify the spatial configurations of the wearer in 3-D space so that absolute location (i.e. student A is in the corner) as well as relative location (i.e. student A is near a toy or near student B) can be used for constructing context information. Furthermore, the speech processing functionality of iBadge enables on-the-spot evaluation of student's progress through speech recognition as well as the ability to provide instant feedback on student's action by sounds or speech from the iBadge or other objects that are located in close proximity. Other supported functionalities include environment sensing, low power wireless communications, and power tracking. Table 1 is the list of constraints on different functionalities that are supported.

Table 1. iBadge design constraints

Localization	position accuracy in 3D space up to 10cm
Orientation and Tilt	degree accuracy up to 10 degree
Speech Processing	processing and recognition of wideband speech
Environment Sensing	requires humidity, barometric pressure, temperature, and light intensity
Wireless Communication	bandwidth larger than 64 kbps for real time speech transmission
Power Consumption Tracking	current tracking accuracy up to 5 mA.

4.1 Localization Unit

Localization Unit is responsible for obtaining precise 3D location of iBadge in the classroom. iBadge estimates its 3D location using an ad-hoc localization process. The

localization process is based on the location knowledge of fixed beacons (node with known location) attached to the ceilings and the ability of the iBadge to accurately measure the distance between itself, the fixed beacons, and other iBadges in the room. To perform the distance measurement, iBadge is equipped with one 40KHz ultrasonic transceiver and a low power radio from RF Monolithics. The distance is measured by recording the arrival time difference of a simultaneous transmission of an RF signal and an ultrasonic burst. A separate radio (RFM TR1000) is needed to supply precise timing information, which is not available from Bluetooth module. Our experimental measurements shown in Figure 4 indicate that iBadge can achieve measurements with a precision of 2cm for distances up to 3 meters away from a beacon.

Using the distance measurements from multiple beacons in the room, iBadge estimates its precise 3-D location. As soon as three or more independent measurements are taken, the location is estimated using an atomic multilateration process [4]. If there is no direct line-of-sight, iBadge may still estimate its position using line-of-sight to other iBadges. Although iBadges cannot estimate their locations individually, jointly they may have sufficient information to estimate their locations using a collaborative multilateration process based on a distributed Kalman Filter [9]. The distributed Kalman Filter also utilizes the inertial measurements from the Orientation & Tilt Sensing Unit to keep track of the iBadge for short time periods when obstacles in the room make ultrasonic distance measurements infeasible.

4.2 Speech Processing Unit

Speech Processing Unit (SPU) consists of a DSP (TMS 320VC5416), a codec chip (TLV320AIC10), a microphone, and a loudspeaker. The DSP is responsible for processing the speech data received from either the codec or AT-MEGA. The SPU on iBadge together with speech recognition software on Sylph are designed to perform distributed speech recognition [13]. Figure 5 shows the principle of two speech recognition scenarios (simple coding using ITU G.722 vs. more complex speech front-end processing) that have been implemented in iBadge and Sylph.

- **Simple coding (top part of Figure 5).** The simple

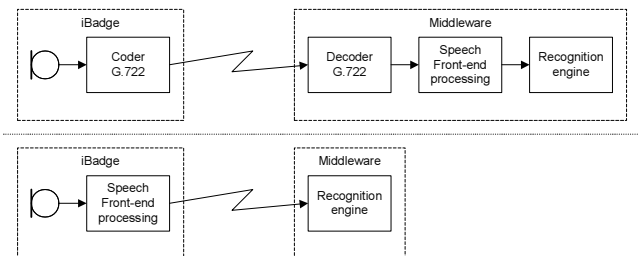


Figure 5. Simple coding vs. front-end processing

coding scenario is used when users want to listen to the raw speech. In this scenario, a codec software runs on the DSP and performs ITU G.722 speech compression. The codec software is implemented in fixed-point arithmetic and has an analog bandwidth of 7kHz. The output bit rate is 64kbit/s, which corresponds to a compression factor of 3.5. We tested the codec on the TIDIGITs database for children's voice. We used the free HTK software (<http://htk.eng.cam.ac.uk/download.shtml>) for the speech recognition process. The setup was the following: The speech recognition engine was trained on the original training speech database. The recognition process was then performed twice on the testing speech database: once, on the original test data and once on the test data, which was first sent through the codec. The recognition rate dropped in general only about 1%, which is shown in Figure 6.

- **Speech front-end processing (bottom part of Figure 5).** The speech front-end processing on iBadge is used when users are just interested in the recognized contents of speech. The entire speech recognition process consists of two parts, the front-end processing and the recognition engine. Front-end processing extracts the speech features from captured speech, which are transmitted to the middleware Sylph where the actual recognition process takes place. The extracted speech features mainly utilize the magnitude spectrum. Because of the loss of phase information, speech can hardly be reconstructed from feature vectors. This constraint leads to an unambiguous strategy. Speech feature extraction runs whenever only plaintext of the spoken words is needed. We process the front-end processing according to the ETSI Aurora specification for distributed speech recognition [13].

Comparing these two scenarios, transmitting speech features has the advantage of a low bit rate of 4.8kbps (compared to 64kbps). However, it needs more computational power to extract these features than just compressing speech as in the G.722 standard. The heavier computational load reduces the energy savings of the smaller traffic. The power consumption difference between computation and communication is a tradeoff that needs to be exploited as mentioned in section 3.2.

4.3 Power Management/Tracking Unit (PMTU)

One unique design feature of iBadge is PMTU, which consists of four MOSFET switches and four battery monitors (DS2438). PMTU provides iBadge with the ability to on-demand switching on/off of functional units in order to conserve power. Moreover, PMTU keeps track of the power that is being consumed by the overall circuit as well as other functional units. The functional units whose power consumption are monitored by PMTU include the Speech Processing unit, the Sensing Units (Localization,

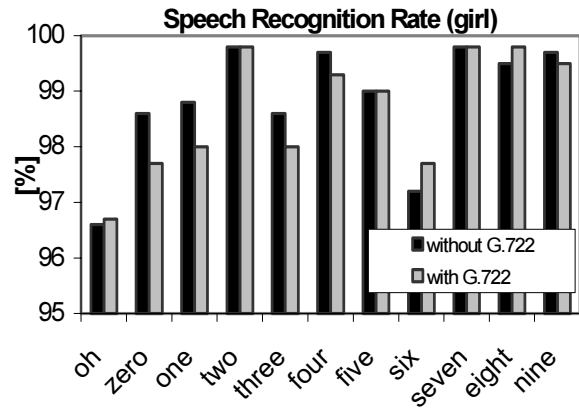


Figure 6. Speech recognition rate plots for spoken digits from 0 to 9

Orientation/Tilt Sensing, Environment Sensing) and the Wireless Communication Unit.

Using PMTU, iBadge can implement power aware algorithms or battery state aware power management schemes to prolong the battery lifetime. Furthermore, iBadge can report the battery state to Sylph, which in turn uses the information to select the iBadge with the longest remaining battery life for heavy power consuming operations.

4.4 Orientation/Tilt Sensing Unit

Two dual-axis accelerometers (ADXL202E) combined with a single tri-axis magnetometer (HMC1023) make up the Orientation/Tilt Sensing Unit. Each type of sensors by themselves can provide separate sensing data of either the student's acceleration (motion detection) or the presence of ferrous material nearby. When combined, these two different type of sensor can act as a tilt-compensated compass which provides accurate Orientation/Tilt of the iBadge. The dual-axis accelerometer is soldered perpendicularly to the iBadge and it provides pitch and roll of the iBadge with respect to Earth's gravity. The magnetometer with X, Y, and Z axis measures the earth magnetic field along each axis which in turn can be used to discover the orientation of iBadge with respect to the earth's magnetic field.

One of the difficulties of measuring accurate orienta-

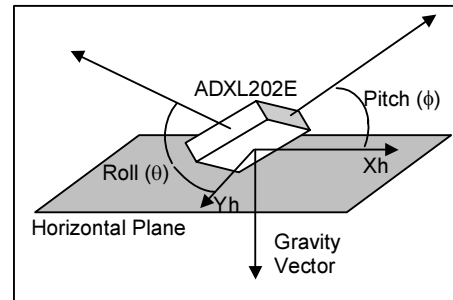


Figure 7. Definition of tilt angles

tion of iBadge comes from the fact that iBadge may not always be in its upright position. With children age 4-6 years old, it is easy to imagine that iBadge will be in various position and tilt based on students' level of activities. If the tilt of iBadge is not compensated, simply reading from the magnetometer can produce large errors in computing precise orientation of the iBadge in 3D space [5]. Figure 7 describes how the accelerometer can be referenced to compensate the X, Y, and Z values read from the magnetometers. Using the θ and ϕ angles obtained from the accelerometer, X_h and Y_h , which are the tilt compensated reading of earth magnetic field parallel to the earth plane, can be computed using equation 7.

$$\begin{aligned} X_h &= X \cdot \cos(\phi) + Y \cdot \sin(\theta) \cdot \sin(\phi) - Z \cdot \cos(\theta) \cdot \sin(\phi) \\ Y_h &= X \cdot \sin(\phi) + Y \cdot \cos(\theta) \cdot \sin(\phi) + Z \cdot \sin(\theta) \cdot \sin(\phi) \end{aligned} \quad (7)$$

5. Sylph design overview

The Sylph middleware shown in Figure 1 acts as a lightweight, extensible proxy service for handling service discovery and mediating access to transducers within the SmartKG. The middleware is designed to provide a common framework for a wide variety of devices using strong abstraction layers. A sensor developer should remain unaware of high-level interfaces, and an application should simply enlist sensors and submit queries.

5.1 Sylph components

To provide these distinct functions, Sylph uses three types of components: sensor modules, which communicate with sensor hardware; service discovery modules, which advertise sensors through some form of lookup service and a proxy core that manages application queries and translates between the two types of modules. A Typical SmartKG session begins with the initialization of the Sylph middleware. Sylph starts by reading in a list of sensor modules and loading the code for each module. Each sensor module may then register its associated devices with the proxy core. New devices can register later or even ask Sylph to dynamically load new sensor modules. After registration, the proxy core uses service discovery modules to advertise each sensor with a specific discovery domain. A discovery service can then direct client applications to the proxy core, which then translates client request to pass through the sensor module interface. When the sensor responds, the core processes returned data for delivery to the client. In the following subsection we look at each of the components in more detail.

5.1.1 Sensor module. The sensor module provides a standard means for initializing and accessing sensor devices including iBadge. When Sylph loads a sensor

module, either during startup or after dynamic sensor detection, the proxy core reads in an XML configuration file, passing a set of parameters to the module. The user can configure a sensor module by modifying simple attribute-value pairs in the file. Sensor modules also implement a common interface for receiving directives from the proxy core and returning data. Both proxy core commands and returned data values take the form of simple attribute-value pairs.

The interface consists completely of commands to SET and GET attributes, which change values and retrieve current values, respectively. For instance, "SET period=1" might set the sampling interval to one second, while "GET period" would return the current sampling interval. Other attributes might only be available for reading, such as a device's remaining battery reserves, or writing (e.g., "SET command=reset" might flush buffers and reboot the device). Many attributes will also have limitations on acceptable input, such as a hardware-limited sampling interval (i.e., the device cannot deliver data any faster) or a restricted command set.

5.1.2 Service discovery module. The service discovery modules allow Sylph to advertise the presence of managed devices to sets of client applications. Sensor modules first register their sensors by providing information in a standard format, including available attributes, data types, and other built-in characteristics. The service discovery module then advertises the new sensors through a standard discovery mechanism. The design of the interface allows the middleware to take advantage of different features provided by various systems, such as soft-state maintenance and attribute searches, but it does not require that all systems provide these features. The service discovery module contract only guarantees that the application can send text-based queries to the middleware and receive data values in response; all other features are optional.

Currently, we have implemented a service discovery module using Sun's Jini discovery service. For each new sensor, the service discovery module creates a stub object and an associated set of searchable attributes, such as the sensor's name and various characteristics of the available data. It then advertises the sensor by sending its stub object to a Jini registrar. We also take advantage of Jini's leasing system, which helps keep registry entries up-to-date by automatically timing out expired leases. This comes at the cost of posting periodic refresh messages to renew a service's lease, but the module can adjust the lifetime of the lease to balance consistency against overhead.

5.1.3 Proxy core module. The proxy core forms the heart of the Sylph middleware, tying together sensor modules and service discovery modules while managing system resources. The core updates data structures to track sensor

devices and user applications, including identifier tags for sensors and other entities in the Sylph system. It also parses incoming requests to form query plans and processing trees. Finally, the proxy core provides an extensible framework for the addition of specialized features, such as mobility support, dynamic query optimization, and quality-of-service control.

A set of unique numerical identifiers allows the proxy core to recognize the various information services and application queries. When a sensor module registers a device, the proxy core assigns it a unique sensor identifier. Similarly, when a client issues a query, the proxy core provides a unique request identifier. In this case, the use of identifiers allows the proxy core to associate data formats with each query without the overhead of data headers. The proxy core even supplies an identifier for itself in order to uniquely identify its own resources and clients from those of other Sylph proxy services (e.g., those that exist in the same Jini discovery domain). Together, these identifiers act as a demultiplexing key, uniquely identifying each endpoint of a network connection. Therefore, the proxy core can provide a small number of common UDP sockets, using the identifier key to direct data tuples to the appropriate internal structures.

5.2 Sylph query language

From the application's point of view, Sylph acts as a flexible data retrieval system. For most purposes, the application need only choose some useful sensors and issue queries for their data. While similar to traditional database systems, the real-time nature of sensor data leads to some important differences. Since sensors can essentially output an infinitely long stream of data tuples, common aggregate functions cannot be applied directly. For instance, to compute an average temperature one would normally sum up all the readings and divide, but there is no fixed number of values in a sensor data stream. Instead, one must use a moving window average (e.g., average just the values in the last hour) or some other approximation. To address these issues, we have developed a query language for the Sylph middleware. Queries specify data values to return, transformations to apply, sampling rates, and durations, or default to built-in settings. A basic request takes the following form:

```
READ <data tuple>
[WHERE <predicate clause>]
[EVERY <time interval>]
[FOR <duration>]
```

<data tuple> is a list of data fields to be returned by the sensor; <predicate clause> is a set of conditions to be met before the sensor will return any data;

<time interval> is the time between samples; <duration> gives the lifetime of the query. Example queries for different components of the iBadge follow, along with explanations:

a) READ temperature, pressure
 EVERY 1 minute FOR 1 hour

This query returns temperature and pressure readings (as ordered tuples) from iBadge's Environment Sensing Unit every minute for an hour. Sylph converts all values to standard types (e.g., milliseconds) for internal processing but allows flexibility in the application interface.

b) READ x, y, z
 WHERE volume > 70
 FOR 30 samples

This query requests the three-dimensional position of the iBadge whenever its microphone records a volume of over 70 decibels (e.g., loud voices or shouting). The query has no expiration time and instead will terminate after recording thirty such events.

5.3 iBadge Sensor Module & Software Interface

We have designed iBadge's Software Interface, which glues together iBadge and Sylph's iBadge Sensor Module. In this section we discuss the high level details of iBadge Sensor Module in Sylph and iBadge Software Interface that resides in ATMEGA. iBadge Sensor Module together with iBadge Software Interface virtualize all the physical sensors of the iBadge. Through virtualization, separate physical sensors can be combined to provide more elaborate sensing services (e.g. an accelerometer combined with a magnetic sensor providing tilt compensated orientation sensing in section 4.4). Furthermore, iBadge Software Interface exports all the accessible sensor services through iBadge Sensor Module.

iBadge Sensor Module translates the request of applications through a standard interface. The interface consists of simple "Get" and "Set" routines (as discussed in section 5.1.1) for each sensor with different configurable attributes. The list of sensor attributes is included in Table 2.

Table 2. Sensor attributes

Variable Attributes	sensor reading, current sampling rate, trigger threshold, trigger state (armed, fired), trigger
Fixed Attributes	sensor ID, data type, max/min bound of sensor reading, unit, max sampling rate, min sampling rate

Based on requests of iBadge's Sensor Module, the iBadge Software Interface configures trigger mode of the sensors to deliver the desired sensor data. The routine for setting up a trigger has the following interface:

```
trid = Trigger( mode, delay, threshold,
               sampling rate )
```

The description of the trigger modes is shown in Table 3. The trigger interface provides an easy way of configuring the sensors with different characters through a uniform interface.

Table 3. Description of trigger modes

trid	Trigger ID; to identify the data with the client
mode	Trigger mode (immediate, threshold >, <, =)
delay	Amount of time iBadge must wait before setting the trigger
sampling rate	Sampling rate once the trigger fires (once, or periodic)

6. Related work

Relevant to iBadge design is the Active Badge project [8] from Xerox Parc where infrared (IR) sensors used to locate the badge's wearer in an office environment. Smart-Badge system [6] from Royal Institute of Technology in Stockholm also includes IR sensors along with environment sensors and speech functionality to offer an environment aware platform that can be used in an office setting. Smart Dust project [7] from UC Berkeley shows a glimpse of the future where tiny sensors with micro-electromechanical systems (MEMS) based technology are massively deployed in the field to constantly evaluate and react to the changes in the environment. It is this vision that is shared by iBadge design where we bring tiny wireless sensors into our everyday life.

The advancement of sensor technologies has driven a wave of research aimed at supporting various sensor services. Researchers at Stanford present an architecture for query processing with data streams in [10], evaluating some of the problems introduced by real-time, infinite data sources, as well as approaches to making real-time data more accessible. For example, Cornell's Cougar project attempts to adapt object-relational abstractions to sensor-directed queries [11]. In support of such database approaches, researchers at UC Berkeley have proposed a system architecture utilizing Fjord data structures [12]. They use composable operators and queues for handling disparate sources and distributing data along multiple paths, similar to the operator trees used in the Sylph proxy core.

7. Conclusion

In this paper we presented the design of iBadge and Sylph in the context of the overall SmartKG architecture. iBadge packages speech processing capability, location sensing, environment sensing, wireless radio, and power monitoring/tracking unit in the size of a small pager that can be worn by pre-school children. In addition, the Sylph

middleware infrastructure provides abstraction layer for initializing and accessing raw physical sensors while employing means of advertising the available sensor services to the higher layers. The main challenge of SmartKG project lies on the successful integration of these two distinct systems, the wearable hardware and the middleware infrastructure. We believe that the wearable architecture of iBadge and the Sylph middleware will play a critical role in assisting parents and teachers with evaluating and aiding students in early childhood education. We also believe that SmartKG will become an essential experimental environment that will enable us to explore novel aspects of person-to-physical world communication paradigm.

8. References

- [1] G. Pottie and W. Kaiser, "Wireless Sensor Networks", *Comm. of the ACM*, Vol. 43, No. 5, May 2000, pp. 51-58.
- [2] M. Srivastava, R. Muntz, and M. Potkonjak, "Smart Kindergarten: Sensor-based Wireless Networks for Smart Developmental Problem-solving Environments", *Intl. Conf. on Mobile Computing and Networking 2001*, Rome, Italy, July 2001, pp. 132-138
- [3] A. Chen, R. Muntz, S. Yuen, I. Locher, S. Park, M. B. Srivastava, "A Support Infrastructure for the Smart Kindergarten: Sylph and iBadge", *IEEE Pervasive Computing Magazine*, vol. 1, number 2, April-June 2002
- [4] A. Savvides, C. C. Han, and M. B. Srivastava, "Dynamic Fine-Grained Localization in Ad-Hoc Wireless Sensor Networks", *Proc. of the Intl. Conf. on Mobile Computing and Networking (MobiCom) 2001*, Rome, Italy, July 2001
- [5] Applications of Magnetic Sensors for Low Cost Compass Systems, Michael J. Caruso, Honeywell
- [6] H.W.P. Beadle, G.Q. Maguire Jr., and M.T. Smith, Using Location and Environment Awareness in Mobile Communications, *Proc. IEEE/IEEE ICICS'97*, Singapore, September 1997, pp. 1781-1785
- [7] J. M. Kahn, R. H. Katz and K. S. J. Pister, "Mobile Networking for Smart Dust", *ACM/IEEE Intl. Conf. on Mobile Computing & Networking*, Seattle, WA, Aug. 17-19, 1999
- [8] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3): 94-104, September 1991
- [9] A. Savvides, H. Park and M. B. Srivastava, "The Bits and Flops of the N-Hop Multilateration Primitive", <http://nesl.ee.ucla.edu/projects/ahlos/reports/TM20020308-1.ps>
- [10] S. Babu, J. Widom, "Continuous Queries over Data Streams," *ACM SIGMOD Record*, vol. 30, no. 3, Sept. 2001, pp. 109-120
- [11] P. Bonnet, J. E. Gehrke, P. Seshadri, "Querying the Physical World," *IEEE Personal Comm., Spec. Issue on Smart Spaces & Environ.*, vol. 7, no.5, Oct.2000, pp. 10-15
- [12] S. Madden, M. Franklin, "Fjording the Stream: An Architecture for Queries Over Streaming Sensor Data," *Intl Conf. on Data Engineering (ICDE 2002)*, Feb. 2002
- [13] ETSI ES 201 108, Speech Processing, Transmission and Quality aspects (STQ); Distributed speech recognition; Front-end feature extraction algorithm; Compression algorithms, 04/2000