An Event-based BSN Middleware that supports Seamless Switching between Sensor Configurations

Christian Seeger, Alejandro Buchmann
DVS Group, TU Darmstadt, Germany
{cseeger,buchmann}@dvs.tu-darmstadt.de

Kristof Van Laerhoven
ESS Group, TU Darmstadt, Germany
kristof@ess.tu-darmstadt.de

ABSTRACT

Recent advances in wearable sensors have surged in novel fitness and preventive health care systems that measure step counts, activity levels, and performed exercises with inertial sensors, enabling users to monitor condition and day-to-day lifestyle. This paper presents a middleware designed for a smartphone unit to support health monitoring applications. Its event-driven architecture enables modular system design and seamless switching between sets of embedded sensors. The strengths of the middleware are highlighted in a deployed feasibility study where daily and gym activities are recognized through an interchangeable set of wireless sensors. The study demonstrates that the setup is suitable for daily usage with minimal impact on the phone's resources.

Categories and Subject Descriptors

D.2.11 [**Software Architectures**]: System architectures—*data abstraction, domain-specific architectures*

General Terms

Design, Performance

Keywords

body sensor networks, middleware, event-based, smartphone, fitness application

1. INTRODUCTION

The World Health Organization predicts that chronic diseases will become the most expensive problem faced by current health care systems and sees the integration of prevention into health care as the main solution for this problem [13]. A paradigm shift towards integrated, preventive health care as well as equipping patients with information, motivation, and skills in prevention and self-management are described as essential elements for solving this problem. As body sensor network (BSN) systems are capable of continuously monitoring a person's physiological and physical state, they form a promising tool that equips patients with the required information and motivation.

Many BSN-based projects in health care [5, 6] focus on monitoring of a particular disease or set of physiological signals. They benefit from the independence from stationary in-hospital observations, allowing patients to freely move and live their daily life while being monitored over longer times and under more realistic conditions. The authors of [3] present the MobiHealth project which consists of a generic BSN for health care as well as a generic mobile health service platform, making a distinction between a basic set of sensors and patient-specific ones such as blood glucose monitors or ECG. The main focus of this work lays on the network infrastructure among a patient's BSN and health care provider, with longer-term patient monitoring as a future goal. In the Partnership for the Heart project [7], 710 patients with cardiac disorders were equipped with a stationary scale, ECG, SpO2, blood pressure sensors, a hip-worn activity sensor as well as a PDA for transmitting the daily measurements to a remote health care provider. Less hospital stays, an increased quality of life, and a faster reaction to health changes are promising results of this study. The system contained a fixed set of sensor and a relatively sparse monitoring technique.

We argue in this paper that support for flexibly handling sensor configurations is a significant feature for many BSN applications. For a patient with a cardiac disorder for instance, monitoring of blood pressure, ECG, and physical activities would be preferred. As monitoring progresses, however, other data might become relevant, such as those from additional respiration and blood oxygen saturation sensors to observe a developing sleep apnea. In another scenario, fitness users might be equipped with basic sensors for activity and heart rate monitoring to log the daily activity level and calorie expenditure, while doing their cardio exercises, additional measurements of respiration, blood oxygen saturation, and ECG might be desired. Our proposal is a BSN system that is designed to seamlessly handle changing sensor configurations, so that introducing new sensors and new sensor rules requires only minimal intervention.

Motivated by these scenarios, this paper presents a BSN middleware that 1) is able to cope with interchanging sets of sensors, 2) is straightforward to deploy, and 3) runs interconnected devices that support day-long monitoring. An event-based system design is combined with modules that translate sensor data to events, to support adapting the system's functionality, extending the sensors set, and cope seamlessly with changing sensors. The middleware is fur-

IHI'12, January 28–30, 2012, Miami, Florida, USA.

thermore designed to run on a smartphone, making use of its connectedness, processing power, and user acceptance.

The advantages are demonstrated in a feasibility study, implementing a fitness diary on top of the middleware. The diary captures the user's activities from a changing set of wearable sensors, and monitors the user's heart rate and energy expenditure, illustrating the base functionality needed for a preventive health care application. Evaluations on real-world deployments show that the event-based middleware supports real-time classification of exercises, changing sets of sensors, and that it can be used continuously without recharging the phone's or sensors' batteries.

This paper is structured as follows: after presenting related work, we will discuss the demands on a generic BSN architecture, especially for health care applications, and describe our design choices in Section 3. Detailed information about the event-based architecture is given and the system is compared to the requirements discussed before. Section 4 describes how the middleware can be used for a preventive health care application. We developed a fitness diary that monitors both the daily activities as well as very detailed activity recognition of 16 gym exercises including repetition count. Based on this case study, we evaluate the system's performance and feasibility in Section 5, after which conclusions and a summary of our main results are made.

2. RELATED WORK

Event-based systems provide a loose coupling of event producers and event consumers. Furthermore, they can cope with a high amount of events, interpret them, and extract meaningful information out of the events. Heterogeneity among event producers and consumers is also supported as well as real-time monitoring of events.

For health care applications, event-based systems have been mainly presented in areas with a high amount of data. Examples are intensive care solutions, real-time sleep analysis, and solutions for establishing large health care networks. All solutions benefit from the efficient data processing provided by event-based systems. In the following we describe two examples.

Intensive care units are equipped with numerous devices for monitoring a patient's health parameters such as heart rate, cardiac rhythm, blood pressure, and many others. Many of them are stand-alone devices with individual alarming systems. Once a patient suffers from a problem, many devices might trigger an alarm according to the parameter they monitor. Instead of alarming multiple problems, a single meaningful event could ease and automate intensive care procedures. The authors in [2] propose an event-based system that combines the events from individual sensors. It integrates in one place historical data, events, rules, and data mining models. Since it is an event-based system it is highly customizable. In addition to this, the system performs data mining for identifying possible future risks (e.g., cardiac arrests).

Besides patient monitoring in intensive care units, the authors of [11] propose an event-based middleware for patient monitoring outside health care facilities. They describe homecare environments as being dynamic and customized to a patient's particular situation. A homecare system sends monitoring reports, and state changes to health care providers and triggers alarms in case of emergencies. By characterizing such a scenario as highly data-driven, the authors chose an event-based system. The particular focus of this work lays on enabling data security by adding dissemination control.

3. THE ARCHITECTURE

This section describes the architecture of our BSN approach consisting of a layered, event-driven middleware and a smartphone as the aggregator device. First a discussion about the requirements for generic BSN solution, especially in the context of (preventive) health care applications, is made. After listing the requirements for such an architecture, our design decisions in response are motivated and details on the implementation of the system are presented.

3.1 Requirements

The examples sketched in the introduction have shown an important requirement on BSN platforms: the network configuration of BSNs can change over time and therefore the system should be easy to adapt to new sensor configurations. Those changes can occur over a long time (e.g., patients getting additional chronic diseases, improved sensors becoming available) or even within a short period of time (e.g., ad-hoc connection to a scale). Many current BSN solutions like the LifeShirt [9] are closed systems with a restricted set of sensors. Adapting them to new requirements is expensive. Since BSN solutions benefit from the advances in new body sensors and the resulting services provided to a user or patient, they should be easy to adapt to new circumstances (e.g., sensors, requirements). Therefore, adaptability, expendability and seamless handling of sensor configuration changes are significant features of a BSN architecture.

We distinguish between two roles of BSN devices: sensor and actuator devices for measurements and physical reactions on one hand, and an aggregator device for collecting sensor data and decision making on the other hand. Sensors and actuators provide events in form of measurement information whereas the aggregator consumes and processes these events. Processing often means to aggregate, store and forward events, but also means to derive new events. A derived event could be the combination of multiple (heterogeneous) sensor readings, or the reaction to combinations of events (e.g., emergency call because of a detected heart attack). In order to provide a history of events, the aggregator should provide sufficient storage capabilities as well as the capability to communicate with other instances (e.g., a health care provider).

The following subsections provide a structured list of targeted requirements that a generic BSN architecture should meet in order to support the variety of applications mentioned in the introduction.

3.1.1 *Networking Requirements*

Protocol Independency. Since network protocols are important for the lifetime, security, and performance of BSNs and might be adapted to a special application, a generic BSN architecture should not rely on a specific protocol. Instead, it is important to support different network protocols and to allow changing them or to support more than one protocol in parallel. This furthermore increases the number and types of sensors the system can handle and hence the applications it can support.

Ad-hoc Connections. A second networking requirement is the ability of seamless switching between sensor configurations. This increases not only the usability of the system but also its opportunities. By enabling a BSN application to utilize sensors as they become available without additional interactions, the application always provides the best possible service. An example is given in Section 4.2: As soon as the user wears the gym gloves, the system utilizes the sensor included in the glove and provides counting the exercise repetitions. Further examples are stationary sensors like a scale or a sensor for medication supply. Those sensors are not always connected to the BSN, but the connection should be established as soon as they become available.

3.1.2 Aggregator Requirements

Central Reasoning. Having an aggregator device which collects all sensor readings simplifies the development and maintenance of a BSN application: sensor fusion and reasoning is done on one device. For changing the behavior of the system or adapting it to a new sensor environment, only parts of the aggregator have to be changed instead of the individual sensors. In addition to this, the body-worn sensors can focus on the sensing process, which simplifies their development and in many cases saves energy.

Modular Architecture. The aggregator device has to be adaptive and extendable in order to support different or changing applications. Therefore, it is important to have a modular architecture running on the aggregator device which allows adding or removing functionality in an easy manner and with minimal impact on the whole BSN system.

Self-configuration. As already mention in the networking requirements, sensor units might appear and disappear during runtime. The aggregator should be capable of adding or removing sensor-related software functionality as sensors appear or disappear. This property of self-configuration ensures best available service based on the current sensor configuration.

Sufficient Local Resources. A requirement on the aggregator's hardware is to provide enough processing power for performing tasks such as sensor fusion and reasoning. As shown in Section 4 the aggregator performs real-time classification of streaming sensor data, coming from multiple sources, and compare what was recognized against other events (such as the current activity to the person's current heart rate). In order to log sensor readings, the aggregator should also provide enough storage capabilities (e.g., storing long-term ECG readings).

Connectivity. Health care applications often require connectivity to remote instances. This could be communication among a patient's BSN and health care providers in order to send health reports or to raise an alarm in case of an emergency. Especially for the latter case, a feedback channel to the patient is desirable in order to be able to react immediately to a serious event (e.g., calming down a patient or verifying that an emergency is taking place). Also, the integration of social platforms and online workout databases for preventive health care require connectivity to remote instances and can lead to an increased user motivation.

3.1.3 Further Requirements

Inter-sensor Compatibility. A sensor might be replaced by another sensor that can be used for the same application but provides the information in another manner. For example, a heart rate sensor might be replaced by an ECG sensor in order to get more detailed information. Since an ECG curve also provides the heart rate and rules for heart rate monitoring might not work with ECG curves, a BSN architecture should provide simple mechanisms to convert sensor information to a system-wide known format even if the information has a higher granularity than the desired one.

Security, Privacy. Since body sensor networks are sensing very personal information, security and privacy are important to keep in mind, even more so for health care applications. In addition to this, system reliability is critical for the acceptance of health care applications.

3.2 Generic Platform for BSNs

After discussing the requirements on a generic BSN architecture, we highlight the reasons why we decided for an event-based middleware running on a smartphone as the aggregator device. Afterwards we describe our architecture in detail and show how it fulfills the requirements defined in Section 3.1. In the following section we will show how to use the generic platform for a specific application.

3.2.1 Event-based Middleware

In our opinion, one of the key properties a BSN middleware has to provide is the support of an arbitrary wide range of different sensors and actuators. They are the key elements for monitoring. In addition to this, a BSN middleware should be able to seamlessly switch between sensor configurations. Therefore, we decided for an event-based BSN middleware, since event-based systems provide a loosely coupled communication and therefore a very flexible multi-sensor and multi-actor communication. It allows ad-hoc sensor configurations as well as seamless switching between those configurations. Another aspect of event-based systems is the creation of event hierarchies, event transformation, and event compositions. These techniques enable our system to adapt events to a given system configuration and to handle future sensors without changing major parts of the system. Converting sensor information from a higher granularity to a lower one is achieved by an event hierarchy. We propose a generic event-based middleware that supports a wide range of sensors and actuators and a seamless switching between BSN configurations. Sensor modules translate sensor-specific readings to sensor events which are then injected into the system.

3.2.2 Smartphone as an Aggretator

In today's society mobile phones are very popular and accepted end-user devices. Especially smartphones with large touchscreens, high processing power, and rich storage and networking capabilities gain in popularity. Since they are powerful and already carried by many users, we chose this technology as our BSN aggregator. A typical smartphone has at least 550 MHz processing power, extendable storage capabilities and it supports different network protocols for Internet communication and at least Bluetooth for BSN communication. In addition to this, phone calls are a helpful functionality for health care and elderly applications: For example, instead of immediately sending an emergency car, a simple phone call could show that the person is fine and a sensor malfunction raised the alarm.

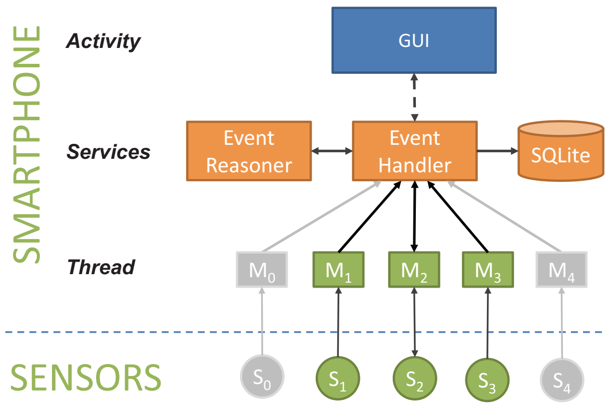


Figure 1: Layered, event-driven middleware architecture for body sensor networks providing high extendibility and adaptability as well as a seamless handling of sensor configuration changes.

3.3 Layered, Event-driven Architecture

Figure 1 depicts the architecture of our middleware implemented as an Android [1] application. On top is the graphical user interface (*GUI*) which is decoupled from the lower levels of the system. It is used for configuration, visualization and user interaction, but the system does not rely on it and runs even if the user is running another application.

The intermediate layer consists of the middleware's main components: the *EventHandler*, *EventReasoner*, and *SQLite* database. They are implemented as services which run independently in different processes which increases both the performance as well as the reliability. In case a service crashes, another service detects this and initializes a re-start. The *EventHandler* is the central communication component that acts like a broker, it consumes events arriving from the sensors, *GUI*, and *EventReasoner* as well as from the application's event producers (cp. Section 4.2). Upon receiving an event, it modifies the event, enriches it or simply forwards it to an interested event consumer such as the *EventReasoner*, *SQLite* database, an application using our middleware or an actuator device connected to the phone. Many events are simply forwarded, but for some applications it is necessary to enrich the event for instance by adding user information (e.g., patient ID, combination of heart rate and current activity). By having event hierarchies, event modifications allow transforming events of one depth in the hierarchy to another depth. For example, an event transformation from an ECG event of high granularity to a heart rate event of lower granularity is very useful if an application is working on heart rate events but not on ECG curves. The *EventReasoner* interprets incoming events, identifies general situations on which the system has to react and creates a corresponding derived event. For instance, the heart rate sensor used in our case study (cp. Section 4.2) provides information about the current battery level. Upon receiving an event from this sensor indication low battery power, an alarm event is created and sent to the *EventReasoner*. The *SQLite* database is used for logging and providing application-specific as well as general sensor information.

At the bottom layer are the sensor modules (M_x). They are running in individual threads and translate the (raw) sensor data to events which are then forwarded to the *EventHandler*. A sensor event basically consists of an event ID, producer ID, producer description, timestamp, and sensor-related information. By having sensor modules as an interface between sensors and the event-based middleware, the system can easily be adapted to new sensors and/or to another communication protocol. For instance, for replacing a heart rate sensor only the corresponding sensor module needs to be replaced. Other parts of the middleware, and even more important, the application itself does not have to be changed. Furthermore, by having the *EventHandler* translating ECG events to heart rate events, even switching from an heart rate sensor to an ECG sensor means only changing the sensor module without changing parts of the application. This feature becomes desirable if new monitoring parameters are added without replacing the old ones (e.g., common heart rate monitoring).

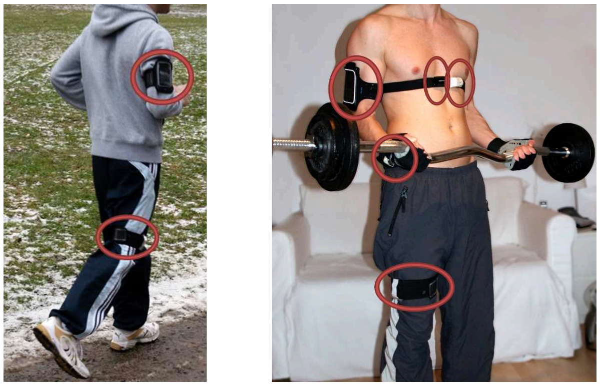
Compared to the requirements on a generic BSN architecture listed in Section 3.1, the following properties are fulfilled by the choice of a smartphone as aggregator and by the previously discussed middleware design:

- Network **protocol independency** is achieved by having modules translating sensor data to events.
- The event-driven architecture supports **ad-hoc connections** inherently.
- **Central reasoning** is done on the smartphone.
- The **modular architecture** provides high extendibility and adaptability.
- **Self-configuration** is achieved by having an event-driven system: incoming events trigger actions.
- Current smartphones provide high **connectivity** and **sufficient local resources**.
- Sensor modules provide **inter-sensor compatibility**.

Having the middleware's and application's services separated in different processes increases the overall reliability. Processes can check for the vividness of other processes. The system's modular architecture simplifies the deployment of **security** and **privacy** features. For securing the sensor network, only the sensor modules have to be changed to the sensor-specific secured communication protocol. A security and privacy module, running on the service layer, can be used for modifying and securing the data before they are sent to a remote instance or for securing the local access.

3.4 Summary

This section showed the requirements on, and our proposal for, a generic BSN architecture. A distinction is made between a fixed aggregator device for handling the communication and the application-related reasoning on one hand. And an application specific set of sensor and actuator devices that might change over time on the other hand. We presented requirements on the aggregator and its communication capabilities. Based on these demands we decided for an event-based middleware running on a smartphone. The resulting architecture was presented in detail and compared to requirements we defined before.



(a) (b)

Figure 2: Body sensor network consisting of a smartphone, a heart rate monitor, and a setup for daily activity recognition (a) and a setup for gym exercise detection including repetition counting (b).

The next section studies the benefits of this architecture in detail and under real-world conditions, by building a preventive health care application, *myHealthAssistant*, on top of this middleware.

4. CASE STUDY: THE FITNESS DIARY

Physical inactivity is one of the risk factors of many costly and disabling health conditions [13]. A goal of preventive health care is to motivate people in increasing their level of physical activity. This case study focuses on motivating a person by continiously capturing daily activities, energy expenditure and the heart rate. Our application, *myHealthAssistant*, is a fitness diary that monitors a person throughout the day and gives, in a standard setup, real-time information about the current activity, energy expenditure, and heart rate. In addition to this, a gym exercise setup gives detailed information about different gym exercises including their repetitions. The captured information could then be shared with friends or sent to a workout database. Activity and heart rate information can be used for calculating the calorie expenditure as proposed in [14]. The calorie expenditure calculation shown in Figure 4 is based on a study from [4] using age, gender, weight and heart rate.

In detail, this case study tests the feasibility of our middleware design and focuses on automated activity recognition that works on different granularities. For capturing a person's daily activity, a coarse-grained activity recognition that detects only a few fitness-relevant activities is sufficient and does only require a small sensor network. For detecting all aspects of a gym workout, more precise activity recognition is necessary and additional information like the repetitions of weight lifting exercises is desired. This fine-grained activity detection needs a larger network of body sensors and increases the complexity of the system. Furthermore, it requires the system to handle changing sensor configurations. The fitness diary recognizes both the coarse-grained daily activities as well as the fine-grained gym exercises including additional repetition information. It stresses different aspects of our architecture such as adaptability, seamless switching between sensor configurations, and multi-modal data processing. Figure 2 shows the sensor configurations of our case study.

Our case study's description is structured as follows: First, a description about system setup including the Android platform and utilized sensor devices is given. Then we show how the application, *myHealthAssistant*, is implemented by using our middleware. Finally, a brief discussion about the application's activity recognition quality is given.

4.1 System Setup

4.1.1 Android Platform

Android [1] is an open source mobile operating system for smartphones and tablet PCs that uses a modified version of the Linux kernel. Software can be written in Java and executed in a specialized virtual machine that provides a wide range of communication protocols. The number and functionality of Android devices grow rapidly and fit very well to the area of BSNs. A smartphone is unobtrusive and, hence, it can be used for daily patient monitoring whereas a tablet PC at the doctor's office can be used for better visualization of the patient's health parameters. Both devices are running the same system and allowing a seamless switching.

The Motorola Milestone phone serves as the Android 2.1 device. It provides 550 MHz processor speed and 8 GB internal storage. The Bluetooth protcol is used for communication with the body sensors. It also supports telephony, text messaging, and different protocols for Internet communication (e.g., EDGE, UMTS, WLAN).

4.1.2 Sensors

The case study consists of up to four sensors connected to our system: one heart rate sensor and three accelerometers. The Zephyr HxM Bluetooth sensor [15] serves as our heart rate sensor which every second sends information about its battery level and the user's heart rate. For the accelerometers we use the Porcupine [12] device which was developed for logging human physical activities. It is small, cheap and equipped among others with the ADXL330 three-dimensional MEMS acceleration sensor and a low-power microcontroller (18F4550). Additionally, we attached a serial port Bluetooth module to it. The sensor's acceleration is processed with a sampling rate of 100Hz. Bluetooth packets containing the following information are sent to the smartphone every second. Each packet consists of the mean values and the variances for each axis over one second of time. Each value is represented by one byte which sums up to six bytes per packet. In addition to this, the wrist sensor adds six bytes peak information to its packet.

The Bluetooth protocol is used for the communication among the sensors and the smartphone since it is well integrated in current smartphones and supported by most Android devices. In addition, there are already various Bluetooth-enabled (health care) sensors as consumer electronics products. This allows establishing a body sensor network using our platform with the extension of a large variety of commercial physiological sensors.

4.2 Fitness Diary Implementation

The fitness diary implementation uses all parts of our architecture described in Section 3.3. A detailed description and evaluation of the application's activity detection and exercise counting algorithms is given in [10]. Figure 3 de-

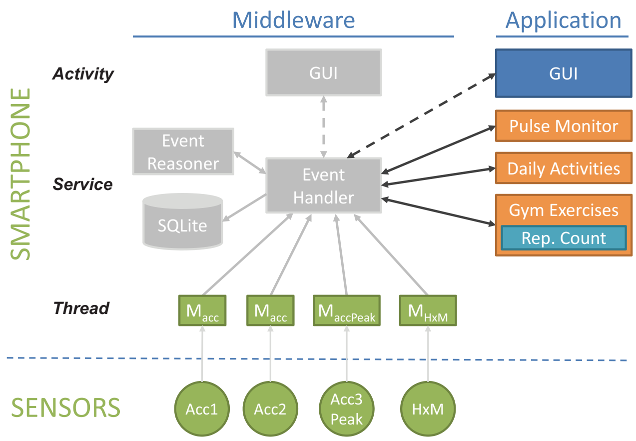


Figure 3: Implementation of *myHealthAssistant*: Application-specific modules send and receive events via the *EventHandler*. Sensor modules translate raw sensor data to events.

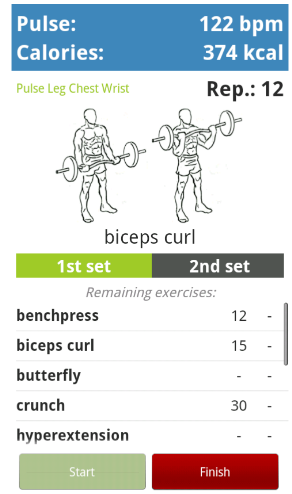


Figure 4: User interface of *myHealthAssistant* showing current heart rate, calorie expenditure, repetition count, exercise, and workout details.

picts the implementation of *myHealthAssistant* with regard to the middleware. At the bottom, there are up to three accelerometers (*Acc1, Acc2, Acc3Peak*) connected to the application, where *Acc1* and *Acc2* are identical sensors providing statistics on acceleration data, and *Acc3Peak* additionally provides peak detection information. The *HxM* sensor represents the Zephyr HxM heart rate sensor. On the figure's right side, the application's user interface and the application-specific modules: *PulseMonitor, DailyActivities* and *GymExercises* are depicted. The modules' descriptions will follow the system's overall information flow.

Starting from the bottom layer, there are up to four Bluetooth sensors connected to the Android phone, namely the *HxM* heart rate sensor and up to three accelerometers (*Acc1* for daily activities and *Acc1, Acc2, Acc3Peak* for gym exercises). The raw sensor data is sent to the corresponding module M_x which translates the data to events. There are three event types created by the sensor modules: *HeartRateEvents* which consist of the current heart rate, an increasing heart rate ID, and the sensor's current battery level, *AccelerationEvents* which consist of the mean acceleration value and variance per axis, and *AccelerationPeakEvents* which additionally consist of peak information per axis. The events are sent to the *EventHandler*. Upon receiving such an event, the *EventHandler* forwards it to modules that are interested in this event type. The following describes the individual modules including their tasks and events they consume and produce.

Daily Activities. The *DailyActivities* module consumes *AccelerationEvents*, sent every second from the three- dimensional accelerometer *Acc1* attached above the right knee (cp. Figure 2 (a)), and performs activity recognition for detecting: standing, sitting, walking, running, or cycling. After calculating the current activity, an *ActivityEvent* is created and forwarded to the *EventHandler*.

The activity detection is based on samples of mean and variance values for each axis on which we modeled the six-dimensional Gaussian distribution for each class. The closest distance to one of these classes of an incoming sample decides on the current activity. This is done in real-time on the phone based on secondly incoming *AccelerationEvents*.

Gym Exercises. For a more detailed detection of the weight lifting exercises, including counting, two more accelerometers are needed: one integrated in a chest-strap (*Acc2*), and the other in a weight lifting glove (*Acc3Peak*) (cp. Figure 2 (b)). All a person has to do is to switch on the sensors, wear them and the system connects to the newly available sensors and begins the fine-grained gym exercise detection. As before, every sensor transmits the mean and variance for each acceleration axis per second, expanding the total input data space for the Gaussian models from six to eighteen dimensions for exercise recognition.

In addition to the *AccelerationEvents* from sensor *Acc1*, the *GymExercise* module consumes *AccelerationEvents* from the sensors *Acc2* and *Acc3Peak*. Based on those events it calculates the current gym exercise out of an exercise space of 5 popular cardio workouts and 11 weight lifting exercises for training the chest, back, shoulders, arms, abs and legs. If a cardio activity is detected, an *ActivityEvent* is created and sent to the *EventHandler*. In the case of a weight lifting exercise, the *RepetitionCount* submodule is triggered and an *GymActivityEvent* including the exercise's repetition count is created and forwarded to the *EventHandler*. Since *GymAcitivityEvents* are inferred from the event type *ActivityEvent*, modules working with *ActivityEvents* can also work with *GymAcitivityEvents*.

Pulse Monitor. The *PulseMonitor* module consumes *ActivityEvents* as well as *HeartRateEvents* and performs a simple pulse monitoring based on the last series of detected activities and the current heart rate. If the current heart rate is outside the activity-specific pulse range, a *PulseAlarm* is created and sent to the *EventHandler*. Besides a "dangerous heart rate" notification on the phone, the system also sends an SMS message to a pre-configured phone number.

GUI. The user interface consumes both types of *ActivityEvents*, *HeartRateEvents*, and *PulseAlarmEvents*. Figure 4 depicts the interface for a barbell curl weight lifting exercise. The current pulse and calorie expenditure are displayed on the top, followed by indicators for sensor connectivity on the left side and the current repetition count on the right side. A picture and the name of the current activity

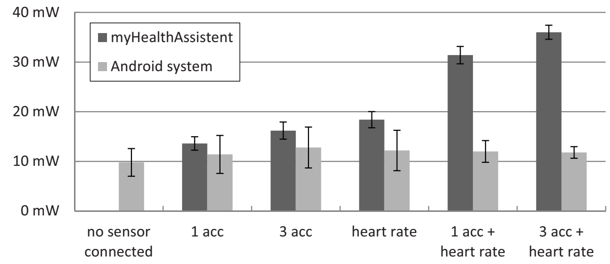


Figure 5: Energy consumption of myHealthAssistant and the Android system with different sensor configurations.

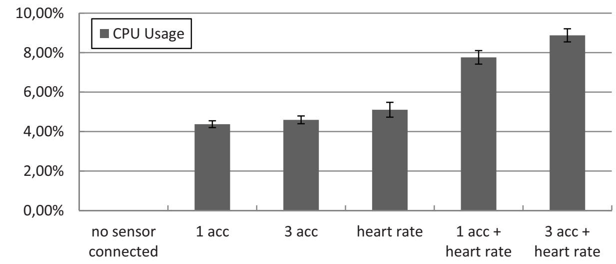


Figure 6: myHealthAssistant CPU utilization of different sensor configurations.

are displayed below this, followed by workout information such as finished sets, performed repetitions, and remaining exercises.

4.3 Summary

The implementation of a fitness diary application built upon the middleware shows that the architecture provides a fast and straightforward application development. Applications are divided in individual modules responsible for specific tasks (e.g., pulse monitoring, activity detection). Modules consume and produce events they receive from and send to the *EventHandler*. For adding new functionality, it is sufficient to add a new module. Event hierarchies allow interoperability among (sensor) modules.

The application's main contribution is an activity recognition that adapts to the environment utilizing the seamless switching between sensor configurations provided by the middleware. Evaluations on the daily activity recognition showed a robust detection even if activities are performed in other speed ranges than they were trained for [10]. Tests on subject-independency also showed a very reliable accuracy of more than 99% for detecting daily activities. This is important for the deployment of a preventive health care application since a user-specific classifier training would take too much effort. For gym exercises, the overall precision and recall are with on average 92% respectively 95% a little worse, but still good enough. Tests on the counting algorithm resulted in an overall miscount rate of 2.42%.

We bundled the *myHealthAssistant* application with the middleware in order to make the deployment as simple as possible. User interaction is limited to installing the Android App and switching on the sensors. For separating the middleware from the application, the application has to connect to the *EventHandler*'s Android Content Provider.

5. SYSTEM EVALUATION

The entire system, with phone and sensors, lasts under realistic conditions (the phone being used frequently, all sensors turned on) for at least 12 hours of activity and heart rate monitoring without a recharge. This is enough time for monitoring a person during the day and charging the system at night. After 12 hours the phone still remains with 20% battery capacity. In future, this can be expected to improve, as the system is not limited to Bluetooth and more power-efficient protocols exist, to which our system can easily be adapted. Figure 5 depicts a comparison of the energy consumption between myHealthAssistant and the Android system for different sensor network configurations using the PowerTutor [16, 8]. The Android system's energy consumption fluctuates during runtime more than myHealthAssistant which might be caused by Android's background tasks. A direct impact of our application to the Android system's energy consumption is not to observe.

A comparison of the CPU utilization between six different network configurations is shown in Figure 6. Without any sensor connected to the application, the application needs almost no CPU cycles. A system consisting of one accelerometer and the activity recognition for daily activities increases the CPU utilization to 4.3%. By connecting two more accelerometers and performing the detection of 16 gym exercises, the system uses almost 4.6% CPU time. The use of a heart rate sensor takes more CPU power (5.1%) because the $PulseMonitor$ is triggered by incoming heart rate events. Thus, connecting the heart rate sensor means that the $PulseMonitor$ starts monitoring the heart rate which also takes CPU cycles. Using both a heart rate sensor and accelerometer(s) for activity recognition increases the CPU utilization again to 7.8% for recognizing daily activities and to 8.9% for recognizing gym exercises. The application requires approximately 22MB memory and 12 hours of heart rate and daily activity monitoring requires 3.8MB in the SQLite database.

Figure 7 shows a day-long test of daily activity and heart rate monitoring. It also shows some unusual positive and negative peaks. The negative peaks were resulting from a too dry contact between the sensor strap and the skin which can be resolved by moistening the strap. For the high peaks, we have not found a proper solution so far. Usually the heart rate returned to the actual value after some seconds. Since the heart rate processing is done on the sensor and only the final value is sent to our system, sensor failures are hard to detect. In our application, the $PulseMonitor$ compares the heart rate against the current activity and an overall range of heart rate values. Since the peak values in Figure 7 were outside these ranges, the $PulseMonitor$ triggered a "'dangerous heart rate"' alarm (cp. Section 4.2) and, hence, informed us about the sensor malfunction.

For gym exercise recognition, sensors have to be attached to three positions on a subject's body (cp. Figure 2 (b)). The heart rate sensor together with one accelerometer is attached to the chest. Since the heart rate sensor's strap is very comfortable and both sensors are small, these sensors are unobtrusive and attaching them is easy to do. The wrist sensor is combined with a weight lifting glove which makes it also easy to attach. Only attaching the leg's accelerometer happened to be slightly difficult for subjects because it is not clear were the sensor has to be. This could be solved

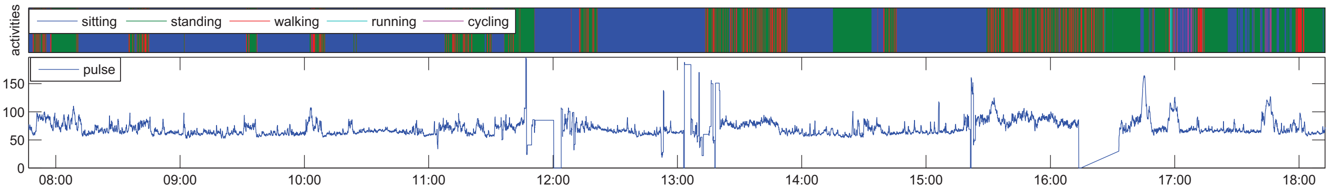


Figure 7: Visualization of the data captured by the BSN system on the phone: detected activities and heart rate values during one of the day-long tests.

by labeling the strap with left/right indicators and showing a sketch illustrating an attached sensor. Once we had shown the subjects how to attach the leg's sensors, attaching it was no more a problem for them. Some subjects had the problem that the strap was slipping during the cardio exercises, especially during a run. For future tests we will try a dimpled rubber strap in order to avoid slipping. Overall, attaching the sensors was fairly easy and readjustments were not necessary.

Despite the peaks from the heart rate sensor, our system worked quite reliably. Only during the long-term tests, the leg's accelerometer disconnected once or twice. Except for the long-term test, there were no Bluetooth disconnections during our tests. The application itself did not crash during our tests.

From an end-user's point of view, the system seemed to be easy and intuitive to use. Except for the leg's accelerometer, the sensors are unobtrusive and wearing the system in a gym did not attract attention. Changing the textual user feedback on the phone to illustrations in form of pictures of the detected activity, resulted in an improved user experience, especially for elderly people.

6. CONCLUSIONS AND FUTURE WORK

Many health care applications require continuous monitoring of patient's physiological and physical parameters. A body sensor network consists of body-worn sensors that allow monitoring a patient's parameters in real-time and therefore fits to those requirements. In this paper, we presented a fitness diary built upon a generic middleware that captures a person's heart rate, daily activities, as well as specific gym exercises. This preventive health care application intends to motivate patients to increase their level of physical activity and to decrease the risk of disabling health conditions.

The contributions of this work is an event-based middleware for body sensor networks, providing the means necessary for supporting a variety of typical BSN applications. Its modular and event-driven architecture supports high flexibility and extensibility as well as seamless switching between sensor configurations, which we argue are important properties for a range of health care applications. In order to evaluate our middleware, we developed a fitness diary application built upon it. The application works on different sets of sensors and shows the middleware's ability to handle different sensor configurations, provide efficient data processing, and day-long monitoring.

Since there are already a lot of commercial health care sensors (e.g., ECG, blood pressure, SpO2) available, our next focus will rely on extending the sensor set to physiological parameter-related applications. In addition to this, enabling communication with the environment and health care providers is another target of our future work. In order to cope with these challenges, we will extend our architecture with additional communication paradigms and integrate mechanism that guarantees security and privacy up to an appropriate degree.

The fitness diary case study is currently in evaluation for further extended use by expert users. Additional integration plans include the uploading of the completed workout to a social platform, in order to illustrate the communication with remote instances and to increase the user's motivation.

The source code of the Android application, the embedded sensor routines, as well as the reported data sets can be obtained by contacting the first author.

7. ACKNOWLEDGMENTS

This work was gratefully supported by the German Research Foundation (DFG) within the research training group 1362 *Cooperative, Adaptive, and Responsive Monitoring in Mixed Mode Environments*.

8. REFERENCES

[1] Google. Android OS. `http://www.android.com/`, 2011. [Online; accessed 31-January-2011].

[2] D. Guerra, U. Gawlick, and P. Bizarro. An integrated data management approach to manage health care data. *DEBS*, pages 40:1–40:2, 2009.

[3] V. Jones, A. Van Halteren, N. Dokovsky, G. Koprinkov, J. Peuscher, R. Bults, D. Konstantas, W. Ing, and R. Herzog. *MobiHealth: Mobile Services for Health Professionals, in M-Health: Emerging Mobile Health Systems*. Springer-Verlag New York Inc, 2006.

[4] L. R. Keytel, J. H. Goedecke, T. D. Noakes, H. Hiiloskorpi, R. Laukkanen, L. Van Der Merwe, and E. V. Lambert. Prediction of energy expenditure from heart rate monitoring during submaximal exercise. *Journal of Sports Sciences*, 23(3):289–297, 2005.

[5] P. Khan, A. Hussain, and K. S. Kwak. Medical Applications of Wireless Body Area Networks. *International Journal of Digital Content Technology and its Applications*, 3(3):185–193, 2009.

[6] P. Neves, M. Stachyra, and J. Rodrigues. Application of wireless sensor networks to healthcare promotion. *Journal of Communications Software and Systems (JCOMSS)*, 4(3):181–190, 2006.

[7] Partnership of the Heart. Telemedical interventional monitoring in heart failure. `http://www.partnership-for-the-heart.de/en/`, 2011. [Online; accessed 01-April-2011].

[8] PowerTutor. A Power Monitor for Android-Based Mobile Platforms. http://ziyang.eecs.umich.edu/projects/powertutor/, 2011. [Online; accessed 23-March-2011].

[9] RAE Systems. Lifeshirt - personal life sign monitor. http://www.raesystems.com/products/lifeshirt, 2011. [Online; accessed 01-April-2011].

[10] C. Seeger, K. Van Laerhoven, and A. Buchmann. myHealthAssistant: A Phone-based Body Sensor Network that Captures the Wearer's Exercises throughout the Day. In *Body Area Networks (BodyNets)*, 2011.

[11] J. Singh and J. Bacon. Event-based data dissemination control in healthcare. *Electronic Healthcare*, pages 167–174, 2009.

[12] K. Van Laerhoven and H.-W. Gellersen. Spine versus porcupine: A study in distributed wearable activity recognition. In *International Symposium on Wearable Computers (ISWC)*, pages 142–149, 2004.

[13] World Health Organization. Integrating prevention into health care. http://www.who.int/mediacentre/factsheets/fs172/en/index.html, 2011. [Online; accessed 01-April-2011].

[14] J. Yang and Z. Liu. Adacem: automatic daily activity and calorie expenditure monitor on mobile phones. In *Proceedings of the Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 409–410, 2010.

[15] Zepyhr Technology Corporation. Consumer HxM. http://www.zephyr-technology.com/consumer-hxm, 2011. [Online; accessed 31-January-2011].

[16] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, CODES/ISSS '10, pages 105–114, New York, NY, USA, 2010. ACM.