Generic processing of real-time physiological data in the cloud

Kevin Lee*

School of Science and Technology,
Nottingham Trent University,
Nottingham, UK
Email: kevin.lee@ntu.ac.uk
*Corresponding author

Kiel Gilleade

School of Natural Sciences and Psychology,
Liverpool John Moores University,
Liverpool, UK
Email: gilleade@gmail.com

Abstract: There is an emerging market in the collection of physiological data for analysis and presentation to end-users via web technologies for applications including health and fitness, telemedicine and self-tracking. As technology has improved, real-time streaming of physiological data, providing end-to-end user feedback has become feasible, allowing for innovative applications to be developed. Currently, there is no standardised method of collecting physiological data over the web for analysis and feedback to an end-user in real-time; existing platforms only support specific devices and application domains. This paper proposes a generic methodology and architecture for the collection, analysis and presentation of physiological data. It defines a standard method of encapsulating data from heterogeneous sensors, performing transformations on it and analysing it. The approach is evaluated through an implementation of the architecture using cloud computing technologies and an appropriate case study.

Keywords: physiological computing; cloud computing; physiology; sensors; frameworks.

Reference to this paper should be made as follows: Lee, K. and Gilleade, K. (2016) 'Generic processing of real-time physiological data in the cloud', *Int. J. Big Data Intelligence*, Vol. 3, No. 4, pp.215–227.

Biographical notes: Kevin Lee is a Senior Lecturer at Nottingham Trent University, UK. He received his BSc, MSc, and PhD degrees from Lancaster University. He was previously a Research Associate at the University of Manchester in the UK, Postgraduate Research Fellow at the University of Mannheim in Germany and Senior Lecturer at Murdoch University in Australia. He has published over 60 papers in the areas of distributed systems and adaptive systems.

Kiel Gilleade is a Consultant and Software Developer in Biofeedback Technologies. He was previously a Research Assistant at Liverpool John Moores University's School of Natural Sciences and Psychology in the UK. He has published in the areas of physiological computing and affective videogame technology and recently co-edited the Springer book *Advances on Physiological Computing*.

1 Introduction

Physiological data can be used to infer the affective, cognitive and physical state of the user, such as athletes using their heartbeat rate to track physical performance (Polar Electro, 2013) This information can be used in a range of applications including health and fitness (Philips, 2013), sports, telemedicine (Fergus et al., 2011), entertainment (Nacke, 2011), affective diaries (Lindström et al., 2006) and self-tracking (Gilleade and Lee, 2011) as well being used in research areas such as of psychophysiology (Stern et al., 2000) and sports science. Traditionally, software supporting the collection, analysis and presentation of physiological data is built for deployment on standalone machines. As technology has improved the types of applications that can be supported has increased, including using the internet to process this data. For example the sharing of physiological data, such as heartbeat rate recorded during exercise regimes, via social networks has become popular (Gilleade and Fairclough, 2010).

The use of the internet allows for various application and research paradigms involving physiological data that were not previously possible or infeasible. For example, the sharing of physiological data via social networks was initially entered in manually by the end-user onto a website. But as technology and methods have improved the process can now be fully automated. One particularly interesting research application that uses the internet to manage physiological data is in the support of ambulatory studies (Wilhelm and Grossman, 2010; Fletcher et al., 2010). Psychophysiological research is traditionally performed under controlled conditions in a laboratory setting in order to observe a given relationship between a subject's physiology and their psychological state. Such controlled experiments, however, are not performed in a natural setting and a subject's response may differ outside of the lab as has been found in stress research (Wilhelm and Grossman, 2010).

Ambulatory sensors allow physiological research to be taken out of the lab in order to capture more normative responses, however this is at the cost of controlling for the situational context. The interpretation of physiological data is limited to certain analyses unless contextual information is collected, for example, if using heartbeat rate in a stress monitoring application, physiological changes resulting from body motion need to be filtered in order to observe for changes relevant to psychological stresses. One solution is to develop a standalone system that captures all the required data streams and process them at source alongside the physiological data. This can be computationally expensive and requires specialised setup. Another approach is to offload processing to the cloud which can dynamically adapt to the load (Lee et al., 2010) as well as providing a means to share the data with other interested parties in real-time. Using web-based services to handle physiological data processing also decouples the data from any specific hardware making it possible to migrate the application front-end to fit its environment, for example, a stress monitoring application could migrate from a desktop to a mobile platform as a patient goes about their day.

There are an increasing number of consumer grade physiological sensors on the market that provide data storage and analysis via proprietary web interfaces. These interfaces are limited in that they service a particular hardware device and restricted data analysis, for example the DirectLife activity tracking service (Philips, 2013) only supports their brand of pedometer. There are web services which provide generic support for physiological data streams data, e.g., Pachube (2013). However, these services are relatively rudimentary and do not provide the level of real-time analysis and presentation options that a mature open service could provide.

The lack of open standards for the real-time collection, analysis and presentation of physiological data limits the support available for the design and implementation of innovative applications. This makes deploying certain types of application problematic, e.g., those analysing multiple live data streams. Furthermore, the lack of standardisation means there is limited market competition for hardware and support software in the physiological monitoring area. Together these factors prevent maturity in physiological monitoring.

Physiological sensor frameworks tend to provide collection, analysis and presentation services on a stand alone machine; migrating these services to the web provides amore usable platform for developing sensor applications. There needs to be a standard, open architecture to support the collection, storage, analysis and presentation of real-time physiological data in as accessible a way as possible for many types of applications. Physiological data is transformable to derivatives used for common analyses (e.g., feature extraction from raw physiological signals) as such there needs to be a way of describing the translations between data formats and analyses. Presentation of the data needs to be able to make use of a multitude of sensors and transformed data. To support real-time applications (e.g., biofeedback stress management), the collection, analysis and presentation workflow needs to be optimised to minimise latency.

This paper argues for the need for a standard, open architecture to support the collection, storage, analysis and presentation of real-time physiological data. It proposes a generic architecture for physiological data processing; this encompasses data collection and storage, analysis and presentation. It aims to enable researchers to rapidly create new physiological applications by just specifying its processing workflow. It also aims to enable researchers from multiple fields to work together on macro physiological projects through a common workflow.

The remainder of this paper is as follows. Section 2 introduces physiological data monitoring, architectures for physiological data processing and analyses existing frameworks. Section 3 introduces a generic architecture physiological data processing. Section 4 presents an evaluation of the architecture through an application case study and deployment in the cloud. Section 5 discusses issues that have emerged from the deployment and evaluation. Finally, Section 6 presents some conclusions.

2 Background

This section provides a background on physiological data monitoring, overview of architectures for physiological data processing using the web and an analysis of existing frameworks for physiological data processing.

2.1 Physiological data monitoring

The human body generates a range of signals that can be captured using sensor technology; examples of commonly captured physiological signals are heartbeat rate, skin conductivity and brainwave activity. This data provides a wealth of information about the user's mental and physical state which can be used to determine, for example, if an individual is stressed, exercising or asleep. Physiological data can be captured in its raw format (e.g., bio-electrical impulses that trigger muscles in the heart) or presented as a derivative of the raw signal (e.g., heartbeat rate).

Physiological data is a real-time signal which can be captured at a variety of sample rates which is typically dependant on the measure being monitored.

Physiological data can be used in a variety of applications including health and fitness, telemedicine, affective diaries (e.g., reliving memories) and self-experimentation (e.g. sleep management). The capture of this data can be automated and streamed over the internet, for example the Body Blogger project (Gilleade and Fairclough, 2010) shares the researchers heartbeat rate on Twitter 24 hours a day. There are several key advantages to capturing physiological data in this manner:

a the data can be submitted for professional analysis, for example sharing blood pressure statistics with your doctor (BP Chart, 2013)

b sharing with friends can provide motivation to achieve a given goal, for example, the sharing of physical effort as measured by heartbeat rate recorded during exercise to help motivate future performance (EA, 2013)

c it can be combined with other data sources to provide mash-up interfaces (Gilleade and Lee, 2011).

The collection, storage, analysis and presentation of physiological data is commonly done on a standalone machine given its application in specialised domains. With the rise in low-cost physiological sensors a number of commercial services have appeared which have migrated the analysis of physiological data and its presentation to web-based services.

2.2 *Architectures for internet-based physiological data processing*

There are a range of commercial physiological sensors used for health and fitness tracking including Polar personal trainer (Polar Electro, 2013), EA Sports Active 2 (EA, 2013), DirectLife (Philips, 2013) and SenseWear (Bodymedia, 2013). Physiological data is asynchronously transmitted to the web service for analysis and presentation. These services are typically tied and limited to a single product, with both the hardware and software being packaged by the same retail company.

Commercial physiological data service providers support the collation of data from physiological sensors and provide a variety of analysis tools. For example, the Microsoft Health Vault (Microsoft, 2013) is an open platform for the collection and storage of an end-user's medical data which can be shared with, for example, other users or the end-user's doctor. Sensors can be added to the service along with data analysis applications. Microsoft Health Vault and similar systems such as Google Health (Google, 2011) are, however, not intended for real-time collection, analysis and presentation of physiological data. FitLinxx (2013) provides real-time capabilities for its physiological data store ActiHealth for any device that supports their BodyLAN interface. Support for new applications is provided by widgets in ActiHealth; these are small self-contained applications hosted on the ActiHealth portal, in a software as a service (SaaS) model. There are also a number of other physiological data service providers that support a range of sensors but are intended for use in a particular application, many of which are aimed at the health and sports domains (e.g., RunKeeper, 2013).

There is also a range of academic developed architectures aimed at supporting internet-based physiological data processing. i-Calm (Fletcher et al., 2010) is both a low-cost/power ambulatory sensor, measuring skin conductance, heart rate and motion, and a web service that provides data collection, analysis and presentation for the device. The i-Calm web service uses IIS Server and an ASP.net front-end client with a MS SQL backend providing near real-time data streaming to end-users. Data can be annotated and also provides an event marker function for experimental situations (via wireless push button). A similar system is The Body Blogger project (Gilleade and Fairclough, 2010) that supports a singular user in streaming real-time physiological data to Twitter.

The EMOListen (Kosunen et al., 2010) system supports real-time streaming of physiological data over a mobile phone for conversion to audio and replay on another device. EMOListen is used to support the deployment of mobile multi-user biofeedback applications. This platform currently supports a range of physiological sensors [e.g., Polar (Polar Electro, 2013)]. Physiological data is streamed via mobile phones to Amazon S3 that translates the input to an audio file. PRESEMO (Chanel et al., 2010) is a similar multi-modal multi-user real-time system but is aimed at augmenting different presentation formats (e.g., workshops, meetings, games) with a visualisation of the user's physiological state.

Open web-based services exist that allow users to push and pull generic real-time sensor data streams using the internet. Pachube (2013) is aimed at capturing environment data (e.g., from sensor motes) through a HTTP API using push/pull requests in which any time series-based data stream can be supported. The contents of a data stream can be described by providing the relevant metadata (e.g., measurement units). A basic graphing service is provided with further analysis and presentation provided via third party apps which pull the data stream.

2.3 *Analysis of existing frameworks*

Existing systems have a number of shortcomings such as device support, analysis techniques and presentation support that prevents the growth and development of physiological data services. The majority of internet services supporting the collection and analysis of physiological data are aimed at single sensor models. Where support is provided for multiple sensor models, the analysis and presentation services are either limited or proprietary preventing validation of experimental approaches. Support for complex real-time analysis and presentation is also limited.

None of the current approaches provide a consistent and open method of transforming physiological data (e.g., from a raw signal to a required derivative). Furthermore, there is no possibility for building pipelines of transformations using standard modules. This makes it hard to use these frameworks to conduct repeatable ambulatory experiments using different sensor technologies and between different research groups. The sharing of experimental conditions in psychophysiological research also tends to be problematic as it depends on authors describing in precise detail their experimental protocols. These frameworks can help the sharing of the experimental protocol to support the repetition and verification of results.

The lack of standardisation, prevents the development of an eco-system of higher level research that depends on a maturing underlying technology. For example the development of ethical and social responsibility frameworks for physiological data monitoring on the internet. The research discussed has shown that there are no generic, open architectures for collection, analysis and presentation of physiological data in real-time on the internet. The following section tackles these issues by proposing a generic architecture for physiological data processing.

3 A generic architecture for real-time physiological data processing

3.1 Overview

As discussed in Section 1 there is currently no standard, open system for the collection, analysis and presentation of physiological data in real-time using web services. Section 2 argued that there is a need for such a system that will allow researchers to rapidly create applications that obtain and utilise physiological data. This section attempts to meet these goals, by setting out the requirements for such an architecture and proposing a generic architecture design to meet these requirements.

3.2 Architectural requirements

A generic architecture that supports the collection, storage, analysis and presentation of physiological data for real-time web services must meet some fundamental goals. There are many types of physiological sensor and many types of processing, including a variety of data translations and analysis that a generic architecture needs to support. Real-time physiological data is often collected for visual interpretation (e.g., biofeedback and medical diagnostics), so supporting multiple presentation applications and integrating with other real-time data sources is vital. For it to be globally applicable, a generic architecture needs to support a common workflow for the collection, analysis and presentation of real-time physiological data. It is useful to define the requirements from the point of view of the two sets of stakeholders that will use this architecture, *application developers* and *end-users*.

- *Application developers* need to be able to define many types of applications involving physiological data such as remote patient monitoring. They should not have to concern themselves with storage, processing or performance elasticity issues; they should be able to focus on domain-specific tasks and the defining of the application parameters.
- *End users* are individuals or groups of people who participate in physiological monitoring activities and upload this data in real-time to online repositories. Depending on the application definition and type, a user may or may not need to visualise their data in real-time and provide context about their actions.

To support both these stakeholders and the wide variety of applications discussed in Section 2.1 a generic architecture must be flexible in its data processing support. The visualisation of physiological data can include real-time, historical and domain specific information. To integrate and prepare this data for further processing or presentation, it can be transformed using computational transformations (e.g., converting from the time domain to the frequency domain).

Physiological data is in one of three transformational states

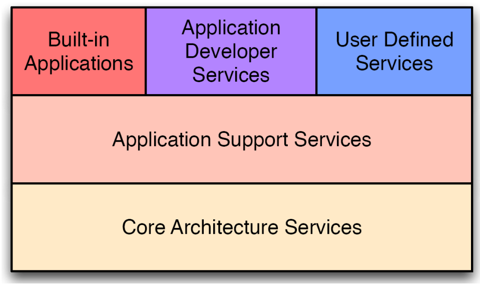
1. *raw* unprocessed data
2. *white box* derivative data which is processed in a transparent manner from the raw or another white box dataset
3. *black box* data which has been processed from a data source using a proprietary technique.

For real-time data presentation an application developer needs to be able to control data transmission properties to require hard, firm or soft timing (e.g., a real-time heart monitoring service for an outpatient with cardiac problems would require firm timing). An architecture supporting different sensor types and analysis techniques will enable the multiplexing of different sensor data and provide real-time multi-modal analysis.

For such an architecture to be viable, it must accepted by the community it is seeking to serve. The user experience must be consistent and enable real-time low latency feedback for end users. Physiological data is inherently personal information, as such, the issue of privacy needs to be supported through end-user preferences. End-users require data input and export facilities to maintain control of their data. For the application developer, the system has to be flexible and become usable long term with minimal maintenance. There are also issues inherent in controlling the interpretation of physiological data (Gilleade and Lee, 2011).

As well as the explicit properties discussed, there are a number of emergent properties associated with the deployment of the system. Having an open, standard repository and workflow will enable researchers to interact on projects. This will enable the crosschecking of experiment methodologies, the development of new approaches to analysis and also the crowd sourcing of the data analysis. The flexibility of a generic architecture will enable new and exciting applications to emerge such as real-time ambulatory physiological monitoring.

Figure 1 The layered approach to physiological data processing (see online version for colours)


Built-in Applications
Application Developer Services
User Defined Services
Application Support Services
Core Architecture Services

The goal of the architecture is to enable a flexible a service model. This is achieved through a layered service model approach as illustrated in Figure 1. The bottom layer provides the core services that allow interfacing between sensors and storage. The application support services layer provides support for different types of physiological data transformations. The three types of applications supported by the architecture are;

1 built-in applications which allow individual users to experiment with collecting data from physiological sensors, perform simple transformations and visualising their data

2 application developer services created by third-party developers

3 user defined services which individual users can create.

3.3 Design

This section discusses the design of a system that implements the requirements of a generic architecture for real-time physiological data processing (GAPP) as defined in Section 3.2. Figure 2 illustrates the relationship between the different entities of the proposed system. To achieve flexibility, it uses cloud services to provide elasticity and flexibility for its processing and storage requirements.

The desirable system structure, as illustrated in Figure 2, is one where there can be many users with each having one or more physiological sensors all collecting physiological data in real-time, transmitting this to cloud-based storage services and retrieving it in real time (along with other domain-specific data) for visualisation. Use of the system by end-users or application developers is through the GAPP system interface, which allows the specification of an application based on its input, transformation and presentation requirements. Applications can be developed that use sensed data together with domain-specific and application specific data stored in the cloud. Applications define what transformations are performed on the data prior to presentation for the user (e.g., converting a raw electrocardiogram signal of volts/per sample to beats per minute, see Figure 9).

The following describe the different stages of usage of the system based on standard web service technologies; XML messages transferred using the RESTful paradigm that use simple HTTP GET and POST between the different components of the system. The system interface supports the management of users and applications, as follows.

3.3.1 User data management

A new end-user can be registered with the GAPP service with a simple XML message, using a HTTP POST, which states the requested user name; as illustrated in Figure 3. The GAPP service performs a simpleMD5 hash of a randomly generated string and returns this to the user as an authentication token along with their user id. This authentication token must be included in all subsequent requests to the service. Once a user has authorised their activity they can participate in the data collection process by registering the required data streams.

Figure 2 The GAPP architecture (see online version for colours)

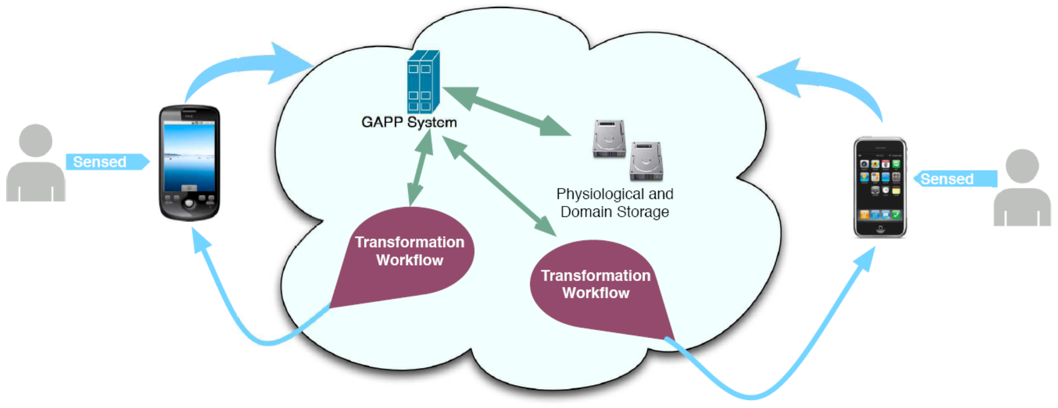


Figure 3 Example user registration message

```
<UserRegistration>
   <UserName>Bob Smith</UserName>
</UserRegistration>
```

The following are the parameters for data stream registration:

- UserID – user's registration ID
- AuthToken – user's authentication token
- Name – name of the data stream
- Description – description of the data stream
- Format – data format definition.

The data format definition describes what kind of data is being uploaded and how it is to be structured. In order to support transformations a data stream must be stored in a consistent format in order to guarantee it can be applied. For a stream involving physical hardware the data needs be bound to a sensor that is done using the following parameters:

- Name – name and model of the sensor
- Description – description of the sensor
- Location – physical location of the sensor.

Location data can be a location on the body or a GPS location as with environmental sensors. Using the properties listed above, a user can register a heartbeat rate sensor with a data stream which provides inter-beat interval information (i.e., time between individual heart beats), this is illustrated in Figure 4.

This message is sent using a HTTP POST and will receive a HTML response or error code in return. A successful message will receive a 200 OK code, 400 bad request, 401 unauthorised, etc. A user has to perform a data registration for each data stream they require to be collected, including physiological, environmental or contextual. For each data registration, the user receives a data registration id that allows the user to submit a data reading.

3.3.2 Data collection

Once a data source has been registered, data can be submitted by building a message consisting of authentication and the following parameters:

- StreamID – the ID returned on a successful data registration
- Timestamp – time the sample was measured (YYYY-MM-DD??HH:MM:SS)
- Fractional – second fractional of timestamp
- Value – sensor reading.

Using these properties, data can be submitted as illustrated in Figure 5. A data submission can contain any number of data readings; single events, collections of data points, aggregate data or derived data.

3.3.3 Data processing and presentation

Data streams made up of data points are processed by defining a series of transformations that operate on an input data stream, perform some processing and produce an output data stream. A user or application developer can specify that instance of a transformation implementation be spawned by the system interface. A transformation is registered using a XML message to the system interface consisting of authentication and the following parameters:

- Name – is the transformation name
- Description – is human readable description
- Input – is the data source(s) for an input(s), specifying the URL, format and StreamID.

Each transformation is instanced as a new web application on the system. The registering user is provided with a web service URL of this instance that can be used to retrieve transformed data from this transformation. Figure 6 illustrates an example of a transformation that converts inter-beat intervals to heart beats per minute for an hours worth of data.

Figure 4 Example user data management message

```
<StreamRegistration>
  <UserID>1</UserID>
  <AuthToken>90eb503faff5501f833532d0e0454312</AuthToken>
  <Stream>
     <Name>Heart Rate</Name>
     <Description>My Heart Rate</Description>
     <Format>IBI</Format>
     <Sensor>
        <Model>BM-CS5</Model>
        <Description>Single ECG channel chest-strap</Description>
        <Location>Chest</Location>
     </Sensor>
  </Stream>
  <Stream>
     <Name>Triggers</Name>
     <Description>Time Index of Events</Description>
     <Format>Trigger</Format>
  </Stream>
</StreamRegistration>
```

Figure 5 Example data submission message

```
<Submission>
  <UserID>1</UserID>
  <AuthToken>90eb503faff5501f833532d0e0454312</AuthToken>
  <Stream>
     <StreamID>3</StreamID>
     <Sample>
        <TimeStamp>2011-06-2T14:08:23</TimeStamp>
        <Fractional>0</Fractional>
        <Value>750</Value>
     </Sample>
     <Sample>
        <TimeStamp>2011-06-2T14:08:23</TimeStamp>
        <Fractional>75</Fractional>
        <Value>1000</Value>
     </Sample>
  </Stream>
</Submission>
```

Figure 6 Example transformation registration message

```
<Transformation>
<UserID>1</UserID>
<AuthToken>90eb503faff5501f833532d0e0454312</AuthToken>
<Name>IBI to HR</Name>
<Description>Convert IBI to HR</Description>
<Input>
    <URL>/GAPP/GAPPServlet?service=getdata</URL>
    <Format>IBI</Format>
       <StreamID>3</StreamID>
    </Input>
</Transformation>
```

Data can be retrieved using HTTP GET requests using simple URL notation as illustrated in Figure 7. The URL of the servlet is called with an additional '?service=getdata' to request data, '&datastreamid=40' to request a data stream, '&starttimestamp=2011-07-13T06:00:00' to specify the starting time stamp and '&endtimestamp=2011-07-13T07:00:00' to specify the end time. An XML message of varying size depending on the number of data entries which fit the criteria of the request is returned to the client. HTTP GET requests also return standard HTTP error codes indicating the success or failure of the request.

Figure 7 Requests the last hour of samples from a data stream

```
HTTP://URL:8080/GAPP?service=getdata&userid=1&
    authtoken=90eb503faff5501f833532d0e0454312&
    datastreamid=3&starttime=2011-07-13T06:00:00&
    stoptime=2011-07-13T07:00:00
```

Transformations are implemented as self-contained web applications. When a user wishes to transform a data stream into another format they must first register the type of transformation they wish to perform with the system. The system creates a instance of that transformation's associated web application and will send the user a unique URL which they can perform HTTP GET requests as they would with a normal data stream. Transformed data can be retrieved by performing a HTTP GET request to the URL of the transformation instance previously registered by the user with the appropriate parameters, as illustrated in Figure 8.

3.3.4 *Transformation workflows*

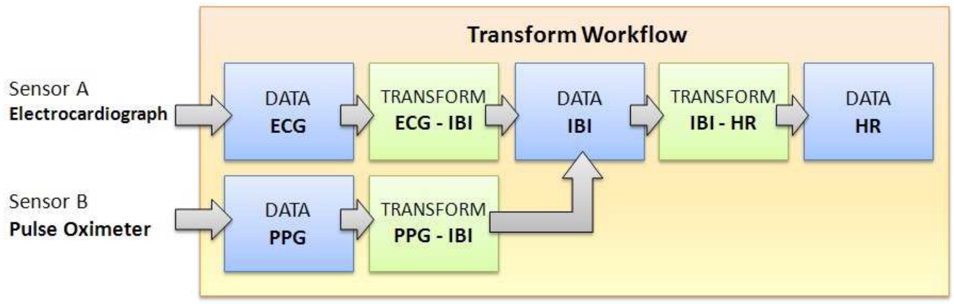
To support complex processing needs, workflows of transformations can be built up in the same way that Grid-based scientific workflow processing is performed (Deelman et al., 2005). Globally available web service URLs are used for the input and outputs to transformation process; the implementation of each transformation handles the processing and storage at each stage. This abstraction allows for flexibility of storage location and also for the dynamic building of transformation pipelines at runtime. Transformation processes can be anything that takes a data input, performs some processing and produces some output; these can be an executable program or a web service as long as they provide a globally accessible web service URL.

The processing performed by each transformation can be applied to simple datasets or complex data streams. The technique used to implement transformations can be simple receive-store-process or stream processing with implications for state maintenance. Continuous query languages can be used to group and process events as data streams using stream query-processing techniques (Arasu et al., 2006) and even perform complex analyse on events streams (Lee et al., 2008).

Figure 8 Requests the last hour of 'transformed' samples from a data stream

```
HTTP://URL:8080/GAPP/Transform001?service=getdata&userid=1&
    authtoken=90eb503faff5501f833532d0e0454312&
    starttime=2011-07-13T06:00:00&stoptime=2011-07-13T07:00:00
```

Figure 9 Transformation pipeline for two types of heart sensor (see online version for colours)



Workflows of transformations can be built up by explicitly specifying inter-transformation dependencies. Figure 9 illustrates an example of using the pipelined approach to transformations to analyse heart rate data. It shows the conversion of different raw signals into a common derivative (i.e., inter-beat interval) that can be further transformed into related derivatives (e.g., heartbeat rate). Many sensors can generate data using different mechanisms but the derivative can still be the same, for example, heartbeat activity can be measured using a pulse oximeter (measures blood flow) or an electrocardiogram (measures electrical activity) from which both an inter-beat interval can be derived from. Transformations in a pipeline don't need to be physically located in the same location; in fact they can be in widely varying locations to take advantage of the cheapest or fastest resources in that moment in time.

In order to create a transformation workflow as illustrated in Figure 9 the user would first register each transformation they wish to perform in turn and pass the returned URL as the input source of the next transformation in the workflow, e.g., the ECG-IBI transform URL would be passed as the input source for the IBI-HR transform. The user would execute the workflow by calling the final transformation in the sequence, which would request data from the previous transformation. This would continue until the transform reaches a data source on the system that would then be passed through the transform chain that has been established.

The approach proposed allows data from many types of data sources to be processed, integrated and presented in a variety of formats in transformation workflows. The presentation can be near real-time on the monitoring device, e.g., a mobile phone, a website for public viewing, or integrated with other data for medical diagnoses. Using transformation workflows, combined client side applications, data from more than one user can be combined for aggregate analysis or presentation. The presentation of this data is left to the application developer. For example, a developer wishing to combine the heartbeat rate data for a group of athletes in order to observe the strenuousness of an exercise regime would

1 register each sensors data
2 have each sensor submit data to the service
3 define a series of transformations to process the data
4 build an application that pulls the output of the transformations and displays this to the user.

3.4 *Summary*

This section has introduced a generic architecture for real-time physiological data processing. It has described the different parts of the system and illustrated how they are used for simple examples. It highlighted a novel technique for performing data processing using transformations exposed as standard web services. The grouping of multiple of these transformations as transformation workflows supports the application developer in creating innovative applications without having to concentrate on technical issues.

4 Case study evaluation: mobile biofeedback interaction

4.1 *Overview*

This section describes a case study-based evaluation of the practical use of the generic architecture for real-time physiological data processing and to illustrate its use in a real-world scenario. Section 4.2 discusses the implementation of the architecture. In Section 4.3 a case study for a prospective real world mobile healthcare application is described. Section 4.4 presents an experimental evaluation using the case study. Finally, Section 4.5 presents a summary.

4.2 *Implementation*

To evaluate the design of the generic architecture for real-time physiological data processing with a suitable case study an implementation of the architecture has been prepared. The architecture has been implemented using open standards-based technologies. This is to ensure maximum compatibility, flexibility of future development and elasticity in responding to application demand. There are three broad areas of implementation concern,

1 the system interface

2 the server back-end

3 application development.; these are now each discussed in turn.

As described in Section 3.3, the interface to the system between the client and server uses standard web service technologies; eXtensible Markup Language (XML) messages transferred using the RESTful paradigm that use simple HTTP GET and POST between the different components of the system. The web service is implemented as a Java Servlet on Apache Tomcat 7 application server (The Apache Software Foundation, 2013) which implements the request-reply programming model using the HTTP protocol. Secure connections can be achieved using HTTPS.

Client applications construct and send XML messages (user registrations, data registrations, transformation registrations and data submissions) using HTTP POSTs to the system interface. All XML messages must conform to specifications in XML Schema Definition Language (XSD) for each message type; both the client and system interface must conform to this. For each XML message sent by the client there is a corresponding transaction confirmation message sent back to the client which confirms the success or failure of the request. In addition, standard HTTP error codes are also used to indicate to the success and failure of any request together with further detail such as a request failing due to an error in the request or an internal server failure.

The server back-end consists of an Apache Tomcat Servlet Version 7.0.16 which utilises a MySQL or SQLite Database. Both the database and application server can be deployed on a standalone PC or deployed using a cloud service such as the Amazon elastic computing cloud (EC2). Using a cloud service such as Amazon EC2 ensures rapid response to computing demand whilst minimising cost when demand is not at its peak.

Broadly speaking, the servlet

1 receives each request from the client

2 validates the client user id and request

3 performs the request by issuing INSERT or SELECT queries to the MySQL database

4 assembles an XML reply

5 returns the reply to the client stating the action performed.

Thus, the servlet is effectively converting XML request-reply messages to SQL queries.

To develop an application, there are three main areas of implementation; data collection, system interaction and data presentation. Data collection involves the sensing of data, formatting it as XML documents and submitting this to the system. As applications communicate using XML over HTTP, client applications are not restricted in implementation language or platform. To interface with the system, the only requirement of an application is that users, sensors and data streams are registered before they can submit data, and that all messages conform to the XSD specifications. Data presentation is left to the application developer, using either the POST or GET interfaces.

4.3 Case study description

For evaluation the following scenario has been selected to demonstrate the ability of the GAPP architecture to process real-time physiological information and distribute it to stakeholders:

- *Scenario*: A patient has been determined to be at risk from heart failure and has been advised by their doctor to increase their daily level of exercise as part of their treatment.. The patient's medical insurance requires that they sign onto an approved health and fitness course, whereby their exercise regime will be managed by a professional fitness instructor, in order to retain their policy.

Low heartbeat variability is a known risk factor in heart failure in chronic heart failure patients (Task Force of the European Society of Cardiology, 1996; Karemaker and Lie, 2000). In treating cardiac patients a portable heartbeat rate monitor can be used to track their condition on a long term basis allowing for the effective deployment of resources when their aliment worsens (e.g., advising patients to return to the doctor). It can also be used to provide feedback on interventions designed to alleviate a condition. In the scenario described above we have three end user's who would benefit from use of the GAPP framework,

1 the patient

2 the patient's doctor

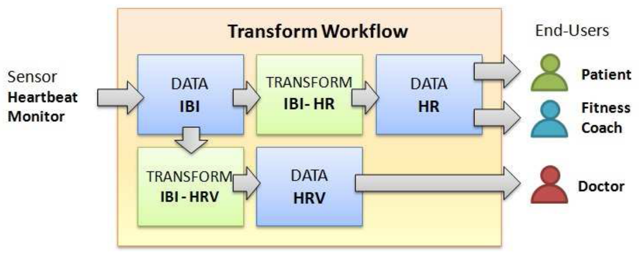
3 fitness instructor.

Each end-user is interested in a measure of heart activity suited to their role in the treatment of the patient.

For both the patient and fitness instructor, heartbeat rate would be a suitable measure. This data format can be used to inform the patient the device is operational and the fitness instructor the patient's level of exertion during exercise. The patient's doctor will be interested in monitoring heartbeat variability as further reductions in variability could be indicative that a condition has worsened and may require a different intervention. Both these measures can be derived from the inter-beat interval (time between heartbeats) which can be collected by a chest-strap-based heartbeat monitor. Heartbeat variability can be assessed using both time (e.g., standard deviation) and spectral measures (e.g., Fourier transform), for the purposes of this case study we are interested in spectral measures which provide information about different cardiac functions (e.g., balance between the sympathetic and parasympathetic nervous systems).

The transformations required by each end-user can be readily supported by the GAPP architecture, the required transformation workflow is illustrated in Figure 10. In this scenario there is a singular input which is processed by two different workflow paths for preparation for the three end-users.

Figure 10 Case study transformation pipeline, preparing heartbeat data for multiple end-users (see online version for colours)



The case study chosen demonstrates the collection, analysis and presentation of real-time physiological data for a health and fitness scenario in which a singular user's heartbeat data is streamed to the GAPP system and is transformed to meet the requirements of three different end-users. In Section 4.4 we describe how the case study is to be evaluated.

This example illustrates a basic single source, multiple output scenario; the GAPP system also supports more complex scenarios. An extension to this case study is one where the system supports multiple patients all being monitored by different fitness coaches and doctors.

4.4 Experimental evaluation

The main considerations for evaluating the real-time physiological data-processing system are

1 can it support realistic physiological application scenarios

2 can it meet the real-time performance constraints of physiological applications and

3 are the GAPP server, transformations and end-user client deployable within reasonable resource constraints.

These are addressed as follows using the defined case study.

4.4.1 Supporting realistic application scenarios

The case study as defined in Section 4.3 describes a remote patient monitoring system which collects, analyses and presents physiological data in real-time to several stakeholders. The described system was implemented using the GAPP architecture.

For maximum flexibility, and to easily support the distributed nature of the case study, this scenario utilises cloud computing services. An Amazon Machine Image (AMI) was prepared using Ubuntu 11.10, Apache Tomcat 7.0.16, MySQL version 14.14 and the GAPP service. This AMI was stored on Amazon Simple Storage Service (S3) ready to be instantiated by EC2. A single instance of this AMI was instantiated on a standard machine type m1.large which has 4 64-bit EC2 compute units and 7.5 GB of memory. Amazon CloudWatch with an AutoScaling group was configured to enable load-balancing and scale the number of instances upon increased CPU utilisation.

An Android-based mobile phone was used to collect heartbeat data from the patient via a Zephyr HxM (Zephyr Technology, 2011), a Bluetooth-based chest-strap-based heartbeat monitor. The timestamp of each detected heart-beat is recorded using an onboard real-time clock with a millisecond resolution. It exposes readings over a serial port as a data packet containing several of the most recent time-stamped heart-beats. This provides for a degree of data recovery in a lossy environment (the slower the heart rate the more data redundancy and vice versa). The inter-beat interval can be derived from this by calculating the difference between heart beat timestamps which can subsequently be used to calculate the user's heartbeat rate and heartbeat variability. The Android phone collects this data using Bluetooth and queries the GAPP web service using the POST and GET interface as described in Section 3.3.

A series of transformations as described in Section 4.3 were registered with the GAPP service, for consumption by the patient, fitness coach and doctor. Client applications for the patient and fitness coach retrieve the output from the IBI-HR transformation using GET requests to the GAPP service. The IBI-HR transformation converts every IBI (in milliseconds) within the requested time period to beats per minute and outputs it as time series in the data submission XML format. Likewise, the client application for the doctor retrieves the output from the IBI-HRV transformation from the GAPP service. The IBI-HRV performs a fast Fourier transform (FFT) on a given epoch of inter-beat intervals. A minimum of five minutes is recommended for this analysis though in less ideal circumstances one or two minute epochs can be used depending on the spectral component of interest (Berntson et al., 1997). For example a two-minute epoch is required as standard for the low-frequency (LF) component (0.04–015 Hz); indicative of activity in both the sympathetic and parasympathetic nervous systems. The LF component is of interest to the case study having been identified as a mortality predictor in chronic heart failure patients (Galinier, 2000). In the case study a two minute epoch is used. As FFTs require an equidistant time series as an input and inter-beat intervals are a non-equidistant time series the data stream must be converted before use which is performed as part of the IBI-HRV transform. The output of the transformation is an XML describing the amplitudes (ms) of the frequency spectra (Hz) in the re-sampled IBI time series.

The flexibility of the GAPP service allows a range of data to be derived and exposed to different end users. The experiment successfully demonstrated that it was straightforward to write simple applications which

1 registers data streams

2 registers transformations

3 allows for the multi-modal retrieval of transformed data in real-time.

4.4.2 Meeting real-time performance constraints

The nature of real-time applications that include physiological monitoring is that they are inherently interactive – requiring live feedback to users on mobile and interactive displays. To support the real-time constraints of physiological data-processing applications the system must have minimal latency. There are two meaningful measurements of latency; the end-to-end transaction time and the latency of each transformation in the transformation workflow chain.

Because of the flexible and lightweight design of the GAPP service, it can be deployed on a standalone machine or remotely on a cloud service with either the MYSQL or the lightweight SQLite database. To test end-to-end latency, the application implementation described in the previous section was deployed and a series of experiments performed.

For the first experiment, 60 seconds of IBIs, derived from heart rate data from the HxM sensor was sent in a burst to the GAPP service. Table 1 illustrates (in the first column) the submission performance achieved. A local GAPP installation achieves an average 423 milliseconds per IBI sample; and a cloud installation, in a realistic scenario using an ADSL connection, performs the submission marginally slower, in 447 milliseconds. Both of are well within the requirement for real-time performance, like, for example, in real-time presentation of the data. Analysis of the system shows that the majority of the overhead is in XML generation and validation. This is the overhead of a generic protocol and can be optimised further.

The second experiment focuses on the latency of performing transformations and retrieving the result. Certain transformations such as IBI-HR operate on the latest data sample, and instantaneous performance is important. Table 1 presents (in the second column) that the end-to-end cost of a IBI-HR transformation, which includes submitting a single data sample and retrieving from the transformation is 958 milliseconds for a local installation and 987 milliseconds for a cloud installation. Performance can be dramatically increased by submitting multiple data points at the same time by buffering them at the source; this will prevent signal drift due to computation delay.

The final latency measurement experiment, which is presented in the third column of Table 1, is the time taken to perform an IBI-HRV transformation on five minutes of IBI data – which is the generally understood minimal useful time period to look for heart rate variability. It shows that it takes 27 milliseconds from a local and 1,544 milliseconds from a cloud installation from the submission of the last IBI to the output of the transformation being received. This shows that the performance of transformations is mainly related to the network bandwidth – due to the output being transferred as a single XML file.

4.4.3 Deployment resource characteristics

Collecting and processing data for physiological applications that use mobile devices and cloud computing services requires using these scarce or costed resources, therefore application footprint is important. There are three locations which the application runs; server, client device and transformation host.

The GAPP service is based on the Apache Tomcat application server, with a Java Servlet performing the application logic. Measurements of the memory footprint on MacOS Lion for the default configuration application server Java Virtual Machine version 1.6.0 with no servlets is on average 71.5 MB including administrative and logging applications.. The addition of the GAPP Servlet increases the Java Virtual Machine memory usage to 81 MB, including SQL Library (MYSQL or sqlite) and XML processing libraries, so the GAPP service has a total footprint of 8.5 MB. A size of 8.5 MB is reasonably compact so can be run on cheap cloud computing services. Memory footprint depends heavily on the OS and Java versions but remains in a similar range.

A GAPP client program simply needs to build the required XML requests and submit these via a HTTP request to the GAPP servlet. The case study described in the previous section uses a simple Java program that builds XML messages and interacts with the GAPP service via HTTP calls. Measurements of the system memory footprint for a simple program including the Java Virtual Machine shows an average of 32 MB. This is the common case, but could be reduced dramatically using an alternative client implementation.

Table 1 Table of performance for case study transformations using local and cloud installations

	Average submission of an IBI sample	*IBI-HR transformation end-to-end latency*	*5 mins IBI-HRV transformation*
Local installation	423 ms	958 ms	7 ms
Cloud installation	447 ms	987 ms	1,544 ms

4.5 Summary

This section has presented an evaluation of the proposed generic architecture real-time architecture for real-time physiological data processing. It introduced an implementation of the architecture and a case study of mobile biofeedback interaction. A qualitative and quantitative evaluation has been presented which shows that the proposed architecture can meet the requirements of wide-ranging and complex physiological monitoring applications.

5 Discussion

The area of real-time physiological data collection, analysis and presentation raises several issues of interest. As physiological data is inherently personal; seeking to collect, store and disseminate this information introduces a variety of privacy and ethical concerns (Gilleade and Lee, 2011). As GAPP is fundamentally a service provider for physiological data transformations in the cloud there is a need to consider the policies governing the type of derivate works that can be produced from the original dataset. In this manner the data owner can apply a level of control over the context under which their data is interpreted.

Beyond understanding that an individual's physiological data is fundamentally personal, collating many individuals physiological data introduces further challenges. With multiple sources of information it is possible to track events and predict behaviour. Data access must be considered, giving access of data to any researcher, outside spammer, insurance company must be reasoned. Ensuring access to data is not misused, must be done by,

1 access to the physiological data feed can be controlled

2 the fidelity of the physiological data can be set to an appropriate level for the service being provided

3 allow the user to control the context

4 allow the user to decide the granularity of the dataset.

Through the user of transformation workflows GAPP can be readily adapted to different processing paradigms. As GAPP uses Tomcat applications to embody unit functions to realise the workflow and distributes these units across the cloud according to load dynamics there is liable to be a trade off between the number of functions in a workflow and its efficacy at executing the entire transformation in real-time. A potential solution for this would be a workflow manager that can distribute unit functions in the transformation according to both the requirements of the transformation workflow, e.g., low latency, and the processing load on the GAPP service.

Expanding upon this paradigm there could be used to support crowd-sourced experiments. As sensing technology has improved it is feasible to replicate studies using consumer grade sensors in a participant's own home. The solution proposed in this paper would enable complex analysis to be performed, at low cost with these consumer grade sensors. In addition, it enables many users of these low-cost sensors to collaborate in social experiments by performing analysis on group data.

6 Conclusions

This paper has argued for the need for a standard, open architecture for the collection, analysis and processing of physiological data to support researchers and end user applications. It has proposed a generic architecture for real-time physiological data processing using cloud computing resources. A case study of patient physiological monitoring was used to demonstrate and evaluation the proposed architecture. It showed that the architecture, and in particular the transformation support, achieves the requirements of the system.

References

Arasu, A., Babu, S. and Widom, J. (2006) 'The CQL continuous query language: semantic foundations and query execution', *The VLDB Journal*, Vol. 5, No. 2, pp.121–142.

Berntson, G.G., Bigger, J.T., Eckberg, D.L., Grossman, P., Kaufmann, P.G., Malik, M., Nagaraja, H.N., Porges, S.W., Saul, J.P., Stone, P.H. and van der Molen, M.W. (1997) 'Heart rate variability: origins, methods, and interpretive caveats', *Psychophysiology*, Vol. 34, No. 6, pp.623–648.

Bodymedia (2013) 'Sensewear armband' [online] http://sensewear.bodymedia.com (accessed 12 April 2013).

Chanel, G., Pelli, S., Ravaja, N. and Kuikkaniemi, K. (2010) 'Social interaction using mobile devices and biofeedback: effects on presence, attraction and emotions', *in BioSPlay Workshop, Fun and Games Conference*.

Chart, B. (2013) 'Blood pressure chart' [online] http://bp-chart.com (accessed 12 April 2013).

Deelman, E., Singh, G., Su, M-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G.B., Good, J., Laity, A., Jacob, J.C. and Katz, D.S. (2005) 'Pegasus: a framework for mapping complex scientific workflows onto distributed systems', *Scientific Programming*, Vol. 13, No. 3, pp.219–237.

EA (2013) 'Ea sports active 2' [online] http://www.easportsactiveonline.com (accessed 12 April 2013).

Fergus, P., Taylor, M., Haggerty, J., Bracegirdle, L. and Merabti, M. (2011) 'Next generation body area networks and smart environments for healthcare', in *Smart Healthcare Applications and Services: Developments and Practices, Medical Information Science Reference*, Hershey, PA, doi: 10.4018/978-1-60960-180-5.ch003, pp.46–74.

FitLinxx (2013) 'FItLinxx' [online] http://www.fitlinxx.net (accessed 12 April 2013).

Fletcher, R.R., Dobson, K., Goodwin, M.S., Eydgahi, H., Wilder-Smith, O., Fernholz, D., Kuboyama, Y., Hedman, E.B., Poh, M-Z. and Picard, R.W. (2010) 'icalm: wearable sensor and network architecture for wirelessly communicating and logging autonomic activity', *IEEE Transactions on Information Technology in Biomedicine*, Vol. 14, No. 2, pp.215–223.

Gilleade, K. and Fairclough, S. (2010) 'Physiology as XP – body blogging to victory', in *Bios-Play Workshop at Fun and Games*. Leuven, Beligum.

Gilleade, K. and Lee, K. (2011) 'Issues inherent in controlling the interpretation of the physiological cloud', in *CHI Workshop on Brain and Body Interfaces: Designing for Meaningful Interaction*, May 7–12, Vancouver, BC, Canada.

Google (2011) 'Google health (discontinued)' [online] http://www.google.com/health/ (accessed 12 November 2011).

Karemaker, J.M. and Lie, K.I. (2000) 'Heart rate variability: a telltale of health or disease', *European Heart Journal*, Vol. 21, No. 6, pp.435–437.

Kosunen, I., Kuikkaniemi, K., Laitinen, T. and Turpeinen, M. (2010) 'Listen to yourself and others – multiuser mobile biosignal sonification platform emolisten', in *Workshop on Multiuser and Social Biosignal Adaptive Games and Playful Applications*.

Lee, K., Murray, D., Hughes, D. and Joosen, W. (2010) 'Extending sensor networks into the cloud using Amazon web services', in *Proceedings of the 1st IEEE International Conference on Networked Embedded Systems for Enterprise Applications, NESEA*, Suzhou, China, November 25–26, pp.1–7.

Lee, K., Paton, N.W., Sakellariou, R., Deelman, E., Fern, A.A.A. and Mehta, G. (2008) 'Adaptive workflow processing and execution in Pegasus', in *3rd Intl Workshop on Workflow Management and Applications in Grid Environments (WaGe08), in Proc. 3rd Intl. Conf. on Grid and Pervasive Computing Symposia/Workshops*, IEEE Press, pp.99–106.

Lindström, M., Ståhl, A., Höök, K., Sundström, P., Laaksolathi, J., Combetto, M., Taylor, A. and Bresin, R. (2006) 'Affective diary – designing for bodily expressiveness and self-reflection', in *Extended Abstracts of CHI*, ACM Press, pp.1037–1042.

Microsoft (2013) 'Microsoft health vault' [online] http://www.microsoft.com/en-us/healthvault (accessed 12 April 2013).

Nacke, L.E. (2011) 'Directions in physiological game evaluation and interaction', in *CHI BBI Workshop Proceedings*.

Pachube (2013) 'The internet of things real-time web service and applications' [online] https://pachube.com (accessed 12 April 2013).

Philips (2013) 'Philips directlife' [online] http://www.directlife.philips.com (accessed 12 April 2013).

Polar Electro (2013) 'Polar personal trainer' [online] https://www.polarpersonaltrainer.com (accessed 12 April 2013).

Runkeeper (2013) 'Runkeeper' [online] https://runkeeper.com (accessed 12 April 2013).

Stern, R.M., Ray, W.J. and Quigley, K.S. (2000) *Psychophysiological Recording*, in Oxford University Press, USA.

Task Force of the European Society of Cardiology (1996) 'Heart rate variability standards of measurement, physiological interpretation, and clinical use', *European Heart Journal*, Vol. 17, pp.354–381.

The Apache Software Foundation (2013) 'An open source JSP and servlet container, Apache Foundation' [online] http://tomcat.apache.org (accessed 12 April 2013).

Wilhelm, F.H. and Grossman, P. (2010) 'Emotions beyond the laboratory: theoretical fundaments, study design, and analytic strategies for advanced ambulatory assessment', *Biological Psychology*, Vol. 84, No. 3, pp.552–569.

Zephyr Technology (2011) 'Bluetooth HxM heart rate monitor datasheet' [online] http://www.zephyr-technology.com/media/pdf/HxMBT-DataSheet-2010-MAR-04.pdf (accessed 12 November 2011).