MusicalHeart: A *Hearty* Way of Listening to Music

Shahriar Nirjon, Robert F. Dickerson, Qiang Li, Philip Asare, and John A. Stankovic
Department of Computer Science
University of Virginia, USA
{smn8z, rfd7a, lq7c, pka6qz, stankovic}@virginia.edu

Dezhi Hong, Ben Zhang, Xiaofan Jiang, Guobin Shen, and Feng Zhao
Microsoft Research Asia, Beijing, China
{dzhong1989, nebgnahz, fxjiang}@gmail.com, {jacky.shen, zhao}@microsoft.com

Abstract

MusicalHeart is a biofeedback-based, context-aware, automated music recommendation system for smartphones. We introduce a new wearable sensing platform, Septimu, which consists of a pair of sensor-equipped earphones that communicate to the smartphone via the audio jack. The Septimu platform enables the MusicalHeart application to continuously monitor the heart rate and activity level of the user while listening to music. The physiological information and contextual information are then sent to a remote server, which provides dynamic music suggestions to help the user maintain a target heart rate. We provide empirical evidence that the measured heart rate is $75\% - 85\%$ correlated to the ground truth with an average error of 7.5 BPM. The accuracy of the person-specific, 3-class activity level detector is on average 96.8%, where these activity levels are separated based on their differing impacts on heart rate. We demonstrate the practicality of MusicalHeart by deploying it in two real world scenarios and show that MusicalHeart helps the user achieve a desired heart rate intensity with an average error of less than 12.2%, and its quality of recommendation improves over time.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems

General Terms

Algorithm, Design, Experimentation

Keywords

Heart Rate, Music, Biofeedback

1 Introduction

Exercise and heart health are so closely related that it is common to see modern workout equipment (e.g., treadmill or elliptical machines), and jogging accessories (e.g. wrist watches or music players) have built-in receivers that continuously receive heart rate information from measurement devices worn on the chest, wrist, or finger. There are, however, many limitations of these heart rate monitoring devices. First, the user has to carry an extra device while exercising, which is an inconvenience. Second, some of these devices require the user to wear a chest strap, which is not only uncomfortable, but also requires extra effort to put on and take off. Third, the best of these devices cost about $400, which is prohibitively expensive for the average person.

We propose a convenient, non-invasive, personalized, and low-cost wellness monitoring system, designed to obtain heart rate and activity level information from a pair of specially designed earbuds while the user listens to music on a smartphone. An intelligent application on the phone uses physiological and activity level information from sensors to recommend, play, and share appropriate music for the user's desired intensity level. The system is *convenient* since the sensors are embedded into the earphone and the user does not have to carry (or worry about forgetting to carry) any additional device. The system is *non-invasive*, since wellness monitoring comes as an additional feature of the earphone that most of the smartphone users already wear in order to listen to music. The system is *personalized* – the user sets his or her personal goal (such as a target heart rate zone) and the device recommends and plays appropriate music– considering his current heart rate, target heart rate, past responses to music, and activity level. The system is *low-cost* – the cost is lower than other standard heart rate monitoring hardware because it is embedded in the earbud.

We offer a complete system comprised of both hardware and software components. We introduce the *Septimu* platform, which consists of a pair of sensor equipped earphones and a baseboard. Both earbuds contain an inertial measurement unit (IMU), an analog microphone and an LED. The baseboard drives the sensors and communicates to the smartphone via the audio jack. An intelligent music player application, *MusicalHeart*, runs on the smartphone and periodically samples the accelerometer and microphone to determine the activity level and heart rate of the person. The information is sent to the server over the Internet, and the server uses it to recommend appropriate music that will help the user maintain their target heart rate. The system is also able to predict the heart's response and enables sharing and streaming of this situation-aware music.

In order to address the challenges of recommending music based on physiological data in real time, we developed three main algorithmic solutions that will be outlined in this paper. First, a fast, accurate and real-time heart rate measurement algorithm is described, which extracts heartbeats from a mixture of acoustic signals from the earbuds. The algorithm is noise resistant, threshold free, and built upon dynamic programming principles to find the optimum solution in real time. Second, a simple yet highly accurate, person-specific, 3-class activity level detection algorithm is described, which leverages the stable accelerometer readings from the earbuds to detect activity levels and posture information in real time. The goal of the activity detector in this work is not to identify specific activity types, but their categories that have an impact on heart rate. However, the algorithm is further enhanced by opportunistically incorporating contextual information – such as the location and speed of the user – from other in-phone sensors. Third, a novel control-theoretic approach for recommending music is described, which constructs a personalized model of each user's responses to different music at different activity levels and uses the model to suggest the best music to satisfy a given heart rate goal.

We design and implement a three-tier architecture for MusicalHeart. Tier 1 is the firmware, which is written in nesC (TinyOS) and runs on the Septimu baseboard. Its role is to drive the sensors, perform on-board processing to reduce data traffic to the phone, and maintain communication with the phone. Tier 2 is the smartphone application, which is written in Java (and native C) and runs on the Android OS. It implements the key algorithms and manages the communication with the server. Tier 3 consists of a set of RESTful web services for extracting and matching music features, music recommendation, sharing, and streaming. We thoroughly evaluate the system components and the algorithms using empirical data collected from 37 participants for heart rate detection and 17 participants for activity level inference. We demonstrate the practicality of MusicalHeart by deploying it in two real-world scenarios and evaluate its performance on 4 users. The main contributions of this paper are:

- *Septimu*, the first wearable, programmable hardware platform designed around low-cost and small form-factor IMUs and microphone sensors that are embedded into conventional earphones, and communicate to the phone via the audio jack.
- *MusicalHeart*, a complete sensing system that monitors the user's heart rate and activity level – passively and without interrupting the regular usage of the phone – while the user is listening to music, and recommends songs based on the history of heart's response, activity level, desired heart rate and social collaboration.
- We devise three novel algorithms: (1) a threshold free, noise resistant, accurate, and real-time heart rate measurement algorithm that detects heart beats from a mixture of acoustic signals from the earbuds, (2) a simple yet highly accurate, person-specific, 3-class activity level detection algorithm that exploits accelerometer readings from the earbuds, and (3) a PI-controller that recommends music to the user based on past history of responses to different music and helps maintain the target heart rate at different activity levels.
- We perform an empirical study by collecting ground truth data of heart rates, and summarizing it to show the effect of music on heart rate at various activity levels. The dataset is further used to show that the detected heart rate is $75\% - 85\%$ correlated to the ground truth, with an average error of 7.5 BPM. The accuracy of the person-specific, 3-class activity level detector is on average 96.8%, where these activity levels are separated based on their differing impacts on heart rate.
- We demonstrate the practicality of MusicalHeart by deploying it in two real world scenarios, and show that MusicalHeart helps the user in achieving a desired heart rate intensity with an average error of less than 12.2%, and the quality of recommendations improves over time.

2 Usage Scenarios

We describe two motivating scenarios of MusicalHeart which are realized and evaluated later in Section 9.

2.1 Personal Trainer: Goal-Directed Aerobics

Alice exercises regularly. Today she wants to practice cardio exercise by going for a jog. She starts the MusicalHeart application on her smartphone and sets her goal to *cardio*. The system initially detects that Alice is standing still and her heart rate is in the healthy zone (50% of maximum rate). The app creates a playlist of songs dynamically and starts playing one that helps Alice to warm up and attain the heart rate for cardio. Alice then starts jogging while the app keeps monitoring her heart rate and activity level. After a few minutes, the app automatically detects that Alice's activity level has changed to a higher level. It dynamically adjusts the playlist according to the new activity level and current heart rate, so that the next song played is suitable for Alice's changing exercise intensity. The system keeps doing this until Alice's jogging session is over. In the end, it adds one more entry into Alice's health journal that keeps her up-to-date about her heart health and exercise history.

2.2 Music Recommendation: Biofeedback and Collaboration

Bob is a middle-aged person who takes the bus on his way home every evening. He enjoys listening to music during this idle time to get some relief after a long tiring day. But he is bored with all the songs that are on his mobile device since he has listened to them at least a hundred times. Today he wants to try some new music, but it has to be something appropriate for the moment– a calm and heart soothing song. He opens up the MusicalHeart application and notices that it already knows about his activity level and context (i.e. traveling). The app connects to the server and obtains a list of melodious and heart softening songs– including some of Bob's favorite titles and novel songs shared by other travelers who enjoy similar music at this activity level. Bob wants to try one of the new titles and the sever starts streaming the song. While Bob is enjoying the new songs, the app is continuously monitoring his heart's responses and reporting the information back to the server for use in future recommendations.

3 Background

In this section, we make the reader familiar with some concepts related to measurement of heart rate, features of music, and heart rate training zones that are used in the paper.

3.1 ECG, Heart Beat and Pulse

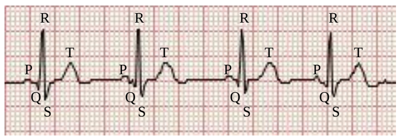


Figure 1. The interval between consecutive R waves determines the heart rate.

The electrocardiogram (*ECG*) is the most reliable tool for assessing the condition of the heart and measuring the heart rate. Getting an ECG reading is a non-invasive procedure in which a set of electrodes is attached to specific regions of a body to record the electrical activity of the heart. Figure 1 shows a portion of an ECG record comprising of four consecutive *heartbeats*. The various waves (or, peaks) that are seen on a heartbeat are historically denoted by the letters: P, Q, R, S, and T. R waves are more visible than others in an ECG record. Hence, the instantaneous heart rate is usually obtained by taking the inverse of the time interval between two consecutive R waves. Since the instantaneous heart rate varies with respiration, the average of $30-60$ instantaneous readings is taken to obtain the average heart rate, expressed in beats per minute (BPM). While the heart rate directly refers to the frequency of the heart's beating, the *pulse*, on the other hand, is the contraction and expansion of an artery due to the beating of the heart. The heart rate and the pulse rate are usually the same – unless for some reason, blood finds it difficult to pass through the arteries. Pulses can be felt on different parts of the body, e.g., neck, wrist, and ears.

3.2 Tempo, Pitch and Energy

As with human heartbeats, the rhythmic, repetitive and recognizable pulsating sound in music, often produced by percussion instruments (e.g., drums), is called the *beat*. The *tempo* of a song is the indicator of the number of beats played per unit time. The tempo of a music is similar to heart rate in human and has the same unit, i.e. beats per minute (BPM). The *pitch* is related to the frequency of the sound wave, and is determined by how quickly the sound wave is making the air vibrate. The *energy* of a signal is computed simply by taking the root average of the square of the amplitude, called root-mean-square (RMS) energy. Figures 2(a) and 2(c) show the time domain and frequency domain characteristics of a 40-second slice of music. Figure 2(b) shows the effect of increasing the tempo and Figure 2(d) shows the effect of increasing the pitch of the music. Increasing the tempo squeezes the samples into a shorter duration, and increasing the pitch shifts the power spectrum towards a higher frequency.

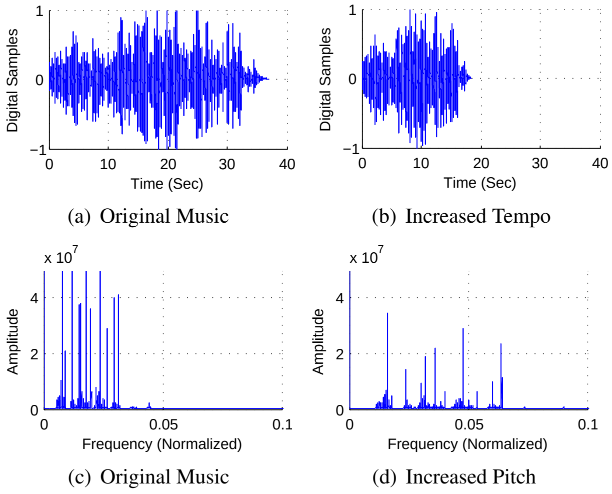


Figure 2. Illustration of tempo and pitch of music.

3.3 Heart Rate Training Zones

Depending on the amount of effort (*intensity*) a person spends during exercise, he or she can be at different heart rate *training zones*. Heart rate training zones are expressed in terms of the percentage of the maximum heart rate of a person. The maximum heart rate of a person can either be directly measured or be computed using Miller's formula: $HR_{max} = 217 - age \times 0.85$. Table 1 shows 5 heart rate zones, the intensity ranges and their corresponding effects. The target heart rate of a person given the intensity value, I, is calculated by: $HR_{target} = HR_{rest} + (HR_{max} - HR_{rest}) \times I$.

Zone	Effect
Healthy $50\% - 60\%$	Strengthens heart, improves muscle mass. Reduces fat, cholesterol, and blood pressure.
Temperate $60\% - 70\%$	Basic endurance. Fat burning.
Aerobic $70\% - 80\%$	Strengthen cardiovascular system. Step up lung capacity.
Anaerobic $80\% - 90\%$	Getting faster. Getting fitter.
Red Line 90%-100%	Working out here hurts. Increased potential for injury.

Table 1. Heart rate training zones.

4 System Architecture

We present a brief description of our system architecture in this section. We describe the hardware platform, software running on the smartphone, and services that run on the server. We defer the algorithmic details to subsequent sections.

4.1 The Septimu Platform

We introduce *Septimu*, which is a redesigned earphone accessory unit for smartphones that enables continuous in-situ wellness monitoring without interrupting the regular us-

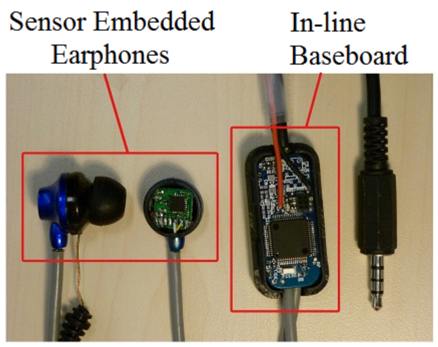


Figure 3. The Septimu hardware platform.

age of the earphone. It is a redesign of the conventional earphone that has additional sensors and communicates with the smartphone via the audio jack interface. Septimu consists of a hardware and a software system. The hardware is comprised of two sensor boards, which are embedded into two earbuds, and a baseboard that collects data from the earbuds and communicates with the smartphone. The current sensor board incorporates an IMU (3-axis accelerometer and gyroscope), an analog microphone and a LED. The baseboard contains a microprocessor (TI MSP430F1611) and peripheral circuits to communicate with the audio jack on the mobile device using HiJack [23]. However, we could not achieve simultaneous communication and power harvesting from the audio jack as described in [23]. We believe this is the limitation of the phones that we tried which are different from the ones (i.e. iPhone) used in their work. Current version of Septimu is powered by a thin film battery. Figure 3 shows the two earbuds and the baseboard. The earbuds, with all the added sensors, has a dimension of $1 \times 1\ cm^2$.

The software running on the Septimu baseboard is based on TinyOS. The microprocessor samples the IMU through I2C bus 25 times per second on each sensor board, with a full scale of $\pm 2g/s$ for accelerometer and ± 250 degree/sec for gyro. Data in the microprocessor is transmitted to mobile phone via microphone tip on the audio jack with Manchester coding, i.e., a logic 0 is represented by a High-Low sequence and a logic 1 is represented by a Low-High sequence. Each byte of samples is sent from the LSB, together with a start bit, a stop bit, and a parity bit. The output is generated at a general IO port on microprocessor and the transmission rate is controllable by software. Septimu communicates with the phone via a digitally controlled multiplexer, to deliver both digital data from the IMU and analog audio data from the microphone in a time-multiplexed manner. The IMU sensor data is reliably transmitted at up to 500 bytes per second while microphone is sampled at 44.1 kHz.

4.2 Processing on Smartphone

The *MusicalHeart* application runs on the smartphone. It uses data from the Septimu sensors to measure heart rate and to detect activity levels. It also uses other in-phone sensors to obtain contextual information. Based on all of this information, it then suggests a list of music to the user. Figure 4 shows a schematic of the software architecture of the application. A complete walk through of the data flow and data processing follows.

1. The user starts the MusicalHeart application on the smartphone which is connected to Septimu. The user may specify a goal, e.g., the target heart rate zone. By default, the system uses the user's current activity level as a basis to suggest music.

2. The application starts a *SensorDataDispatcher* service which generates 4 sensor data streams. The first two streams correspond to the IMU and the microphone units of Septimu. The other two streams correspond to the in-phone GPS and the WiFi scan results. The GPS and WiFi being power hungry are sampled once per minute and more frequently only when the user is detected to be moving. The streaming rate of the Septimu IMU is 50 Hz, while the maximum sampling rate of the microphone is 44.1 KHz.

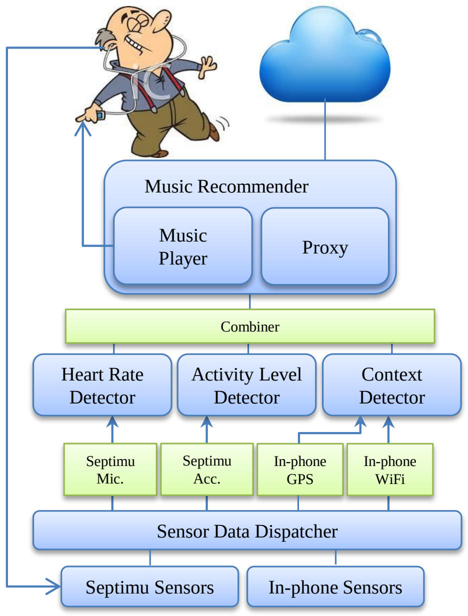


Figure 4. System diagram of MusicalHeart.

3. The sensor streams are consumed by 3 processing units. The *HeartRateDetector* component processes the Septimu's microphone data to obtain the heart rate using the algorithm described in Section 5. The *ActivityLevelDetector* component processes the Septimu's IMU data to obtain the activity level of the user. The GPS and WiFi data are processed by the *ContextDetector* component to obtain contextual information, such as the location (indoor vs. outdoor) and speed of the user. The details of the activity level and context detection algorithms are described in Section 6.

4. The *Combiner* component combines the processed data obtained from the three processing units into a time series of 3-tuples (Heart rate, Activity Level, Context). It maintains an in-memory circular queue to hold the tuples that are generated during the last 10 minutes. Thus the Combiner possesses the high level information of the user, i.e., what activity level the person is in, what the heart rate is, and any contextual information such as whether is at home, or traveling.

5. The *MusicRecommender* is responsible for suggesting music to the user and playing the song. It has a music player which is an extension of an ordinary music player, but with the added capability of loading and playing songs based on biofeedback. It kicks in once prior to the end of a song and consults the Combiner to get user's information during the last song. It computes a report consisting of the heart rate information (start, end, minimum, maximum, average, fall time, and rise time), activity level (low, medium and high), contextual information (location, and velocity), and the user specified goal. All this information is sent to the server via a proxy. The proxy is responsible for maintaining communication with the server and obtaining the suggestion for the next song to be played.

4.3 Web Services

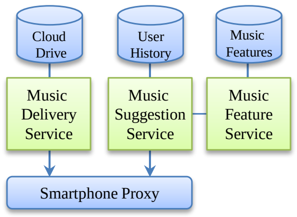


Figure 5. Web services in MusicalHeart.

There are several advantages for storing and serving the *MusicSuggestionService* on the Web. If the information gathered by the Combiner exists in a shared location, the system can leverage the learned parameters from how people will generally react to a particular song offering better suggestions in the future.

To implement this, we pushed the MusicSuggestionService onto a RESTful Web Service that runs on a web server as shown in Figure 5. Our particular implementation uses a WEBrick server to expose a HTTP interface to the smartphone. During each song, the Combiner reports the user authentication, song ID, current heart rate and activity level information to the MusicSuggestionService in a POST request. Next, important features related to the song are fetched from a Music Information Retrieval database. There are several tools, e.g., openSMILE, MIRToolbox, jAudio that can extract musical features to create a custom feature database. However, we use the EchoNest web service that already has the data for 30 million songs. When an artist and title of a song is given, a report of the song's tempo, pitch, energy, loudness, and musical mode is returned as a JSON object. This information is used by the MusicSuggestionService to make a prediction for the appropriate song to play when a particular target heart rate is requested. The details of the algorithm is described in Section 7. When a song is selected, it is delivered to the smartphone either from the user's cloud drive, or is played from a local MP3 file on the smartphone.

5 Heart Rate Measurement

A tiny microphone is embedded inside Septimu which utilizes an in-ear design. This forms a resonant chamber inside the ear, amplifying the sound of heart beats. We have chosen acoustic sensing because of its potential use as a clinical stethoscope. Other sensing methods, such as IR-sensors, do not offer such opportunities and thus limit our possibilities. However, we show a comparison of the acoustic and IR-based heart rate detection techniques in Section 8.4.

Heart beat detection algorithms [17, 31] that are used to detect R waves in an ECG do not work for our problem due to the varying nature of the received signals and the presence of noise. ECG waves are stable and uniform in nature, and ECG is performed in a very controlled environment such as a hospital. In our case, the received signals differ in shapes and sizes as different people have different sizes of ear canals. The received signal also depends on how tightly the earbuds fit in someone's ear. Furthermore, our system is designed to support detection of heart beats even when the person is engaged in high levels of activities. These call for a new algorithm that is accurate and robust to detect heart beats from acoustic signals from the ear.

The raw audio samples that we collect from the microphone is a mixture of the music, the heart beats, human voices and other background noise. Measuring the heart rate from the audio samples is a two-step process – *(1) Filtering*: separating the heart beats from other signals, and *(2) Detection*: identifying the R waves and measuring the heart rate.

5.1 Filtering

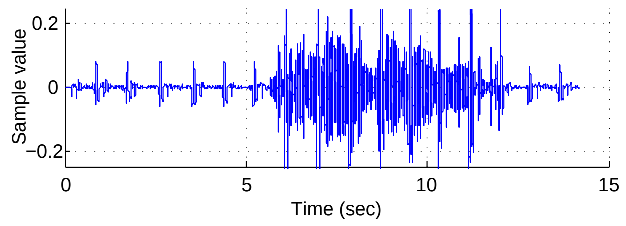
Our heart beats in a specific rhythm. The resting heart rate of an adult person lies somewhere in between $40-100$ BPM, and the rate may reach up to 220 BPM during exercise. To extract the signals corresponding to the heart-beats, we eliminate any signal whose frequency is higher than the cutoff frequency, $f_c = 3.67$ Hz, corresponding to the maximum 220 BPM. Since the microphone sensor has a sampling frequency, $f_s = 44.1$ KHz, the normalized cutoff frequency of the low-pass filter is calculated by: $W_n = 2 \times \frac{f_c}{f_s} = 1.66 \times 10^{-4}$. In our implementation, we use a second order Butterworth filter. The filter coefficients, $a = [1.0000, -1.9993, 0.9993]$ and $b = 10^{-7} \times [0.0680, 0.1359, 0.0680]$, are obtained from the standard chart for Butterworth filters and are plugged into the standard difference equation to filter out the unwanted signals.

Figure 6 illustrates the effect of applying the filter on a mixture of music and heart beat signals. The duration of the experiment is 15 seconds and the presence of music is from $6-12$ seconds. We see that the heart beat signals are clearly visible after the filtering. Due to some higher order harmonics of the music, the heart beat attains a constant gain. But this does not affect heart rate since the gain is only in the amplitude of the signal which does not affect the rate.

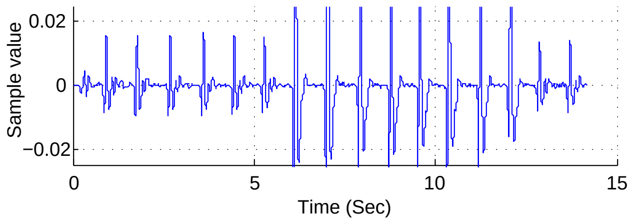
5.2 Detection

The heart beat detection algorithm takes an array of signal amplitudes with timestamps as inputs and returns the positions of the detected beats. The algorithm is applied on a fixed sized time window of 10 seconds. The smartphone accumulates 10 seconds of signals and the following steps are performed to detect the beats:

1. Selection: A set of candidate R waves is selected by taking the peaks that are at least one standard deviation larger than the mean. R wave being the largest, this step practically never misses any of them.



(a) Before filtering



(b) After filtering

Figure 6. Heart beats are extracted from the mixture of heart beats, music and noise by low-pass filtering.

2. Matching: Each of the candidate R waves is matched to its nearest S wave. Since an R wave is a local maxima, the next local minima is the corresponding S wave. We denote the set of RS pairs with χ.

3. Quality Assignment: Each RS pair, $x_i \in \chi$ is assigned a real number, $q(x_i) \in [0,1]$, which is its normalized peak-to-peak distance. The closer the value is to unity, the more likely it is to be an actual heart beat.

4. Dynamic Programming: Given the timestamps, $\{t(x_i)\}$ and the quality values, $\{q(x_i)\}$ of all the candidate RS pairs, our aim is to find the subset, $\{x_i\} \subset \chi$, which contains all (and only) the actual heart beats.

There are $2^{|\chi|}$ possible subsets that could be a possible solution. In order to find the optimum one, at first, we quantify the candidate solutions using two metrics based on the following two properties of the optimum solution:

Property 1: The variance of the time differences, $\Delta t(x_i, x_{i-1})$ between two consecutive R waves, x_i and x_{i-1}, in the optimum solution, is minimum.

Property 2: For two solutions with the same variance, the optimum one has the larger sum of quality.

The first property comes from the fact that the heart beats are periodic and there is hardly any change in the heart rate within the 10 seconds time window. Our empirical study in Section 8.3 shows that it takes about $25-50$ seconds to see any noticeable change in heart rates. The second property ensures that we take the beat sequence with a smaller period, which has the larger sum of qualities. Hence, we define the following metrics for $X \subset \chi$:

$$Q(X) = \sum_{x \in X} q(x), \text{ and } V(X) = \sum_{x_i, x_{i-1} \in X} \Delta t(x_i, x_{i-1})^2 \qquad (1)$$

A brute force algorithm to find the optimum solution is to compute $V(X)$ for all $2^{|\chi|}$ subsets, and take the one that has the minimum $V(X)$ while breaking any tie by choosing the one with the largest $Q(X)$. But this is impractical, since even for a 10 second window, the size of the search space may exceed 2^{36} in the worst case. Instead, we simplify the search by dividing the search space. Each of the subspaces corresponds to a particular value of heart rate within the range of $40-220$ BPM. The search is performed using the following dynamic programming approach.

Let us assume, we know the heart rate in advance. This gives us $|X|$, i.e., the number of actual heart beats in χ, and also sets the minimum time interval between two consecutive beats. We define $f(i,k)$ as the cost of selecting k beats out of the first $i \le |\chi|$ candidates. We now apply the following recurrence equations to find the optimum value of $f(|\chi|,|X|)$, which is the $Q(X)$ for a given heart rate:

$$f(i,k) = \begin{cases} \max\limits_{1 \le j \le i} \{q(x_j)\}, & \text{if } k=1, i \ge 1 \\ \max\{f(i-h,k-1)+q(x_i), & \\ f(i-1,k)\}, & \text{if } 1 \le k \le i, i \ge 1 \\ -\infty, & \text{otherwise} \end{cases} \qquad (2)$$

Here, h ensures the minimum time interval constraint; it denotes the number of previous candidate beats to skip in order to maintain the given heart rate. Once we have the optimum solution we compute $V(X)$ using the selected beats. We solve the above recurrence once for every heart rate values in the range $40-220$ BPM, and the one that gives us the minimum $V(X)$ is the heart rate of the person. Ties are broken with the largest $Q(X)$. Note that, if we only maximize $Q(X)$ without considering $V(X)$, the algorithm will greedily choose all candidate R waves; the job of $V(X)$ is to discard the false peaks that are out of rhythm.

Time Complexity: Let, s, c, and b be the number of samples, candidate beats, and actual beats, respectively. The complexity of selection and matching steps is $O(s)$, quality assignment is $O(c)$, and the dynamic programming is $O(cb)$. Since $c \ge b$, the overall theoretical worst case time complexity is $O(s+c^2)$. However, for a 10 seconds time window, we have $6 \le b \le 37$, which is essentially a constant. A candidate beat that is not an actual R wave is either a high peaked T wave that immediately follows an R wave, or the shadow of the R wave (called R' in medical literature). The wave form being periodic, assuming c a constant multiple of b, the overall time complexity is practically $O(s)$.

6 Activity and Context Detection

A pair of IMUs are embedded inside Septimu, which allows us to infer the activity level of the user. Combined with existing sensors on the phone, we are able to detect the context of the user which enables more suitable music recommendations.

6.1 Activity Level Detection

The activity level is an important indicator of what kind of music a person may prefer. For example, when a person is jogging, which is a high activity level, he may prefer music with faster rhythms. On the contrary, when a person is sitting and is at rest, he may prefer slower music. A person's activity level is considered to suggest appropriate songs. We use acceleration collected from the earbuds to detect the wearer's activity levels, which are divided into three categories: (1) L_1: low level, such as lying and sitting idle; (2) L_2 : medium level, such as walking indoors and outdoors; and (3) L_3: high level, such as jogging. We use standard machine learning techniques to distinguish the activity levels.

6.1.1 Feature Extraction

Given the raw 3-axis accelerometer data from the Septimu sensors, we extract the feature values using the following steps:

Step 1: For each 3-axis accelerometer data sample, (a_x, a_y, a_z) from the ear buds, we calculate the linear magnitude of acceleration: $|a_{ear}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$.

Step 2: The standard deviation of $|a_{ear}|$ within each 1 second time window (i.e. 50 samples) is calculated – which is our feature value, x_i.

6.1.2 Training and Classification

Instead of using a generic threshold, we train an unsupervised learner so that our activity level detection is person specific. We use k-means clustering algorithm to cluster $\{x_i\}$'s to find out the 3 cluster means corresponding to 3 activity levels.

Given the 3-axis accelerometer data from the Septimu sensors for T seconds, the steps classifying the activity level are described as follows:

Step 1: We obtain a set of T values, $\{x_i\}$, one value for each second, following the same procedure as the feature extraction stage.

Step 2: Each of these x_i's is then classified considering their minimum distance from the cluster heads obtained during the training.

Step 3: We take a majority voting to determine the most likely activity level of the person over the T seconds duration.

In our implementation, we choose the value of T to be 60 seconds, because a more fine grained activity level detection is not necessary in our application. We are concerned about the user's activity level during the duration of a song. If a user, for example, stops for moment to tie his shoelaces during jogging, we classify the entire duration as L_3 activity.

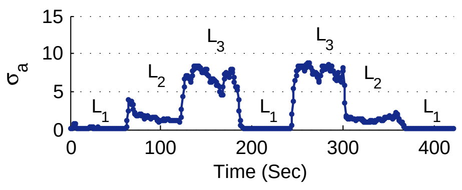


Figure 7. The standard deviation of accelerometer readings are used to distinguish among the activity levels.

As an illustration of the algorithm, in Figure 7, we plot the standard deviations of accelerometer readings at each second during a 7 minute long experiment performed by one of our participants. The same trend is observed when we tested our algorithm on 17 participants; see Section 8.5. The user performs this sequence of activities: sitting, walking, jogging, sitting, jogging, walking, and sitting. We see a clear distinction among the 3 activity levels.

6.2 Augmenting Activity Levels with Contexts

Adding contextual information along with the activity levels makes the system more intelligent in choosing music. For example, sitting at home and sitting inside a bus are both level 1 activities. But if we can differentiate between these, we can suggest music based on the context – e.g. what music is listened to by other users during traveling, vs. when they relax at home. Since there are infinite possibilities, we conduct a survey on 208 people, asking one simple question – *when do you listen to music the most?* The users were allowed to mention at most 5 contexts. The summary of responses results in a list of the 7 most common contexts and is shown in Table 2.

Context	Description	Poll Result
LIE	Lying on bed	14%
SEATED	Sitting idle, or taking a break at desk	41%
TRAVEL	Traveling by bus, or car	24%
SLOWMOVE	Waiting at bus stop, walking on campus	7%
BIKE	Biking	2%
GYM	Exercising at gym	6%
JOG	Jogging or running	4%

Table 2. Activity contexts and corresponding poll result.

We augment activity levels with 3 types of information that helps to distinguish among the 7 activities: indoors/outdoors, velocity, and head angle (Septimu accelerometer's pitch and roll angles). Table 3 presents the context detection technique in a tabular form.

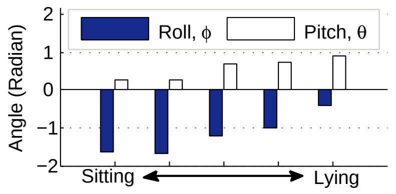


Figure 8. The roll and pitch angles obtained from Septimu are used to differentiate between sitting and lying.

Context	Activity Lev	In-Out	Velocity	Roll, Pitch
LIE	L1	Indoor	-	$> -1.5, > 0.5$
SEATED	L1	In/Out	-	$< -1.5, < 0.5$
TRAVEL	L1	Out	$> 25\ mph$	-
SLOWMOVE	L2	In/Out	$< 3\ mph$	-
BIKE	L2	Out	$< 15\ mph$	-
GYM	L3	In	-	-
JOG	L3	Out	-	-

Table 3. The context detection algorithm is shown in a tabular form. Dashed entries are don't cares.

To implement this, we detect whether a person is indoors or outdoors, and measure his velocity, roll and pitch angles. We rely on Android's location service to periodically obtain the location updates from its location providers once per minute. The pitch and roll angles of the accelerometer data is used to distinguish between lying and seated contexts. Figure 8 shows an example scenario where one of our participants lies down straight from a seated position. The earbuds being stably situated, we observe the same trend in all 17 participants. Based on the empirical measurements, we use

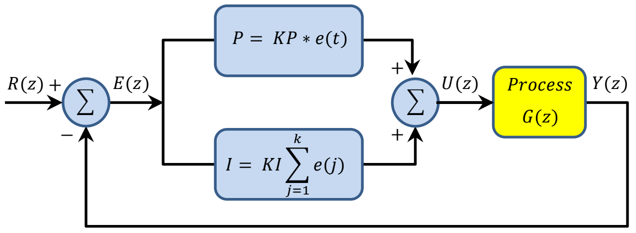


Figure 9. The PI controller uses the desired and current heart rate to suggest appropriate music.

static thresholds of -1.5 for roll and 0.5 for pitch angles. The thresholds are fairly conservative and unlikely to make errors unless the earbuds are deliberately worn in an unusual way (e.g. upside down).

7 Music Suggestion and Rating

We describe a feedback control theoretic approach for suggesting music to the user. Existing approaches [29, 30, 12] apply simple rules of thumb to suggest music, i.e., *suggest the next song with a higher (or lower) tempo if the heart rate is falling behind (or rising above) the desired rate*. Although none of these works are built upon any sound mathematical basis, typically they mimic proportional controllers (P-controller) when viewed in the light of control theory. These controllers therefore suffer the same inherent problems of any P-controller. They are too responsive to the control input (harmful for human as the heart has to respond very fast to the changed music), have non-zero steady state error (harmful because the heart rate may settle at a different rate in the steady state), and have no memory of the past (does not consider the effect of heart's response to the previous song).

7.1 PI-controller Design

We propose a Proportional-Integral Controller (PI Controller) in order to achieve the desired heart rate in human by correctly selecting the music. A PI-controller is a nice fit to our problem since it has a slower response time than a P-controller, a minimum steady state error, and takes the past errors into account. These are desirable properties of our music suggestion algorithm as they provide a slower change in heart rate which is comfortable for the user, attain the desired heart rate with minimum error, and consider the errors that are made in the previous stages. We now describe the design of the controller in detail.

7.1.1 Model

Figure 9 shows a schematic diagram of the PI-controller. The process having the transfer function $G(z)$, represents the part of the human heart that is sensitive to music. The input to the process, $U(z)$, represents the suggested change in the feature of music, the output, $Y(z)$, represents the current heart rate, $R(z)$ represents the desired heart rate, and $E(z)$ represents the difference between the two rates. K_P and K_I are the coefficients of the P- and I-controllers that we are going to compute. The transfer function of the PI-controller is:

$$\frac{U(z)}{E(z)} = \frac{(K_P + K_I)z - K_P}{z-1} \tag{3}$$

And the transfer function [20] of the feedback loop is:

$$F_R(z) = \frac{[(K_P + K_I)z - K_P]G(z)}{(z-1) + [(K_P + K_I)z - K_P]G(z)} \tag{4}$$

7.1.2 System Identification

System identification is the empirical procedure to model the empirical transfer function of the system being controlled. In our system, we model it by empirically estimating the change in heart rate when there is a change in control input which is a function of the features of the music. We assume a first order system with the transfer function:

$$G(z) = \frac{A}{z-B} \tag{5}$$

Taking the inverse z transform of Eq. 5, we get the system response in time domain:

$$y(k+1) = Au(k) + By(k) \tag{6}$$

We define u as a linear combination of 3 features of music: tempo (f_1), pitch (f_2), and energy (f_3), and express it by: $u = \sum \alpha_i f_i$. It can be thought of as a *composite feature* of a song that has linear effect on changing the heart rate. The coefficients, α_i are estimated by applying linear regression on the empirical heart rate responses from *all users*. While the definition of u is generic, the values of A and B parameters are *person specific*. For each user, these parameters are recomputed by the server after each use of the application. When a user finishes listening to a song, the server computes $u(k)$ of the song, and adds $u(k)$, $y(k)$, and $y(k+1)$ into the person's history. This history is then used to estimate A and B using *least squares* regression.

7.1.3 Design Goals

There are four design goals that need to be specified in designing any controller: stability, accuracy, steady state error and overshoot. These are summarized in Table 4. The stability of the controller is achieved when the poles of Eq. 4 are within unit circle [20]. We allow a steady state error of 5 BPM. The settling time of 1 unit means the duration of a single song, and we allow a maximum overshoot of 10%.

Goal	Requirement
Stability	Poles of F_R are inside unit circle.
Accuracy	Steady-state-error < 5 BPM
Settling Time	$k_s \le 1$
Overshoot	$M_P < 0.1$

Table 4. Design requirements of the PI-controller.

7.1.4 Characteristic Polynomials

Using the desired properties of the system from Table 4, we compute the desired poles, $e^{\pm j\theta}$, of the system as:

$$r = e^{-4/k_s} = 0.018 \tag{7}$$

$$\theta = \frac{\pi \ln(r)}{\ln(M_P)} = 5.458 \tag{8}$$

The *desired* characteristic polynomial having these two poles is:

$$(z - re^{j\theta})(z - re^{-j\theta}) = z^2 - 0.025z + 0.00034 \tag{9}$$

The *modeled* characteristic polynomial as a function of K_p and K_I is obtained by substituting $G(z)$ in Eq. 4 and taking the denominator of the system transfer function, $F_R(z)$:

$$z^2 + [A(K_P + K_I) - (1+B)]z + (B - AK_P) \qquad (10)$$

Equating the desired polynomial to the modeled polynomial, we solve for K_P and K_I to obtain:

$$K_P = \frac{B - 0.00034}{A}, \text{ and } K_I = \frac{0.9755}{A} \qquad (11)$$

7.2 Automated Music Rating

The server only has knowledge of the user responses to a song if that song has ever been listened to by the user. For each such song, a personalized rating is automatically computed by the system. A song gets different ratings at different activity levels and desired goals based on the steady state BPM error that it makes. We rate each song in a linear scale of $0-5$, where 5 corresponds to BPM error < 5, and 0 corresponds to BPM error of 20 or more.

For the songs that the user has not yet listened to, but some other user of MusicalHeart has, we estimate their expected rating by considering the similarities among songs and other user's ratings. Given a set of rated songs $\{s_i\}$, and their corresponding ratings, $\{r_i\}$, the expected rating of an unknown song, s_u at the same activity level with the same goal is computed by:

$$\hat{r}(s_u) = \alpha \sum r(s_i) p(s_i, s_u) + (1-\alpha)\bar{r}(s_u) \qquad (12)$$

Where, $p(s_i, s_u)$ is the Bhattacharyya coefficient [4] which computes the similarity score between two songs, $\bar{r}$ represents the average rating of the song by other users, and α controls how much we want to rely upon other users' feedback. Each song is represented by a 3 element feature vector corresponding to the tempo, pitch, and rms-energy of the music. The value of α is a function of amount of rated song by a user. Initially, α is set to 0.1, and is linearly incremented up to 0.7 when the number of rated songs exceeds 100. Hence, in the long run up to 30% of the suggested songs come from other user's responses.

8 Technology and Algorithm Evaluation

We describe three sets of experiments in this section. First, we evaluate the performance of Septimu hardware and firmware. Second, we describe and summarize the findings of our empirical study which is used as the empirical dataset for other experiments. Third, we describe a set of experiments where we use the empirical data to evaluate the performance of the heart rate measurement, activity level detection, and context detection algorithms.

8.1 Experimental Setup

The experiments are performed using multiple Android Nexus S smartphones that run Android OS (Gingerbread 2.3.6). The devices use WiFi 802.11 b/g connectivity and are equipped with a 1GHz Cortex A8 processor, 512 MB RAM, 1 GB Internal storage, and 13.31 GB USB storage. Note that, Septimu is generic hardware platform that does not require any specific operating system such as Android. Septimu even works with desktop computers without any change. It is just that our particular implementation of MusicalHeart is done on Android for a demonstration. The web server run on a computer running Ubuntu Linux 11.10 with 4Gb of RAM and a AMD Phenom X3 processor.

8.2 Evaluation of Septimu Platform

We measure the maximum data rate of Septimu by sending a known bit pattern ($0xA5A5$) from the MSP430 to the smartphone at varying rates and then calculating the bit error rate (BER) at the receiving side. We keep transmitting data for 120 seconds at each rate. We observe that Septimu has almost zero (10^{-5}) bit error up to 5.56 kbps, but the BER keeps rising at higher rates. The limitation comes from the limited processing ability of MSP430. However, in our system, we use a data rate of 3 kbps which is sufficient for our application and ensures minimum error.

We measure the power consumption of Septimu hardware and the android mobile device with a power meter [3]. The Septimu consumes 42 mW power in a battery powered case. For the mobile device, the power measurements are summarized in Table 5. The estimated battery-life of a smartphone (having a 1500 mAh battery) with Septimu connected is about 22 hours.

State	LCD	Audio	Septimu	Power
Standby	off	off	disconnected	< 15 mW
Home Screen	on	off	disconnected	420 mW
app	off	on	disconnected	200 mW
app	off	on	connected	250 mW

Table 5. Power draw breakdown of the application running on an Android phone (Samsung Nexus S).

We obtain the memory usage of Septimu firmware from the compiling information in TinyOS. Out of 10 KB ROM and 48 KB RAM of the MSP430, Septimu uses only 5.8 KB of ROM, and 288 bytes of RAM.

8.3 Empirical Study

We perform an empirical study involving 37 participants, in which, we measure their heart rates during three different levels of activities, and both with and without the presence of music. The activity levels corresponds to {lying, sitting idle}, {walking indoors and outdoors}, and {jogging}. The number of participants in each activity level are 17, 10, and 10, respectively. The participants listen to different types and numbers of music items during multiple sessions. Each of these sessions is $10-60$ minutes long. The durations of the songs are about $2-5$ minutes and there are about $1-5$ minutes gaps in between two songs. The collection of songs comprises of the most watched 100 songs in year 2011 on YouTube. The songs vary in genre, language, length, and volume level. The group of participants is comprised of undergraduate and graduate students, researchers, professionals, and their family members. Their ages are in the range of $20-60$, and they have diversities in sex, physical structure, and ethnicity. The properties of the music are obtained using MIRtoolbox [26].

In order to obtain the ground truth of the heart rate, we use one 3-lead ECG device [2] and one pulse oximeter [1]. Both of the devices measure and store heart rate data at a rate of 1 sample per second. While an ECG is the most reliable method to measure the heart rate, this is not always convenient, as we have to attach three electrodes on to the bare chest of the subject. Specially, for outdoor experiments, we require another means to obtain the ground truth. We use a fingertip pulse oximeter for this purpose, which is worn at the index finger and the instantaneous heart rate is measured and stored in its internal storage.

8.3.1 *Heart Rate: Rise and Fall*

It is a well-known fact that music has a profound effect on our heart [22, 15]. We are not therefore trying to reprove their relationship; rather, our objective is to study the effect of music on smartphone users who are possibly mobile and therefore at different levels of activities. We are interested to know, e.g., how much is the effect of music in rising or dropping one's heart rate, how long it takes to make the effect and whether these have any relation to the activity levels. In Figure 10, we answer these questions from our empirical observations. Figure 10(a) and 10(b) show the amount of rise and fall in heart rates at different levels of activities – both in presence and in absence of music. We obtain these plots by first dividing the songs into two classes – one that tends to raise the heart rate and the other that slows down the heart. This distinction is made by computing the duration of the largest rise or the largest fall of heart rates. We do this after taking the 10 seconds moving average to rule out any transient behavior in the instantaneous readings. We observe that, the rise in heart rate is more in the presence of music and also the fall of heart rate is less when music is playing. The higher the activity level, the more effective it is in raising the rate or resisting the rate to drop. However, this rise and fall in heart rate is not instantaneous. Figure 10(c) shows the cumulative distribution function (cdf) of the rise and fall time of heart rates. A rapid growth in cdf happens when the time is in between 25 – 50 seconds. On average, the time for heart rate to rise or fall by at least 10% is about 50 seconds. Figure 10(d) shows the amount of the rise and fall. The longer a person listens to the music, the more is its effect. For example, listening to an exciting music for 5 minutes raises the heart rate by 10 BPM, while a soothing music for 5 minutes calms down the heart rate by 5 BPM on average.

8.3.2 *Effect of Tempo, Pitch, and Energy*

Tempo, pitch and energy are three important features of music that correlate to changes in heart rate [15, 14]. But their effect varies with activity levels. For example, tempo is more effective in raising the heart rate when a person is jogging or exercising in rhythm with the music. On the other hand, pitch and energy have greater effect than tempo during the low activity level. This is why we introduce the composite music feature u, which is a linear combination of these three and is activity level specific. Figure 11 shows the correlation of tempo, pitch, energy and the composite feature, u with heart rate. We observe that, u is more correlated than the other 3 features and is consistent over different activity levels.

8.4 Evaluation of Heart Rate Measurement

We evaluate our heart rate measurement algorithm using two datasets. The first dataset contains the empirical

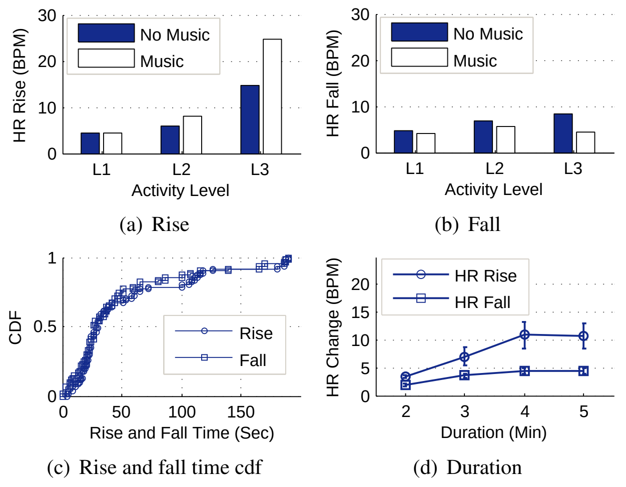


Figure 10. Rise and fall of heart rates.

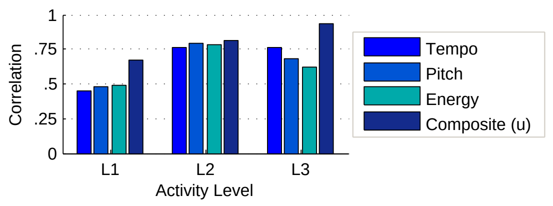


Figure 11. The composite feature is more correlated to heart rate change at all three activity levels.

data that we collected from 37 smartphone users. The other one is the MIT-BIH Arrhythmia dataset [18] which is the most used dataset for evaluating heart rate measurement algorithms, and contains data from 48 real patients. We also compare our acoustic-based technique with an implementation of IR-based technique.

8.4.1 *Performance on Empirical Dataset*

Figure 12 shows the performance of our heart rate measurement algorithm when applied to the empirical dataset. Figure 12(a) shows the BPM errors of MusicalHeart when compared to the baseline, i.e., pulse oximeter in most cases. At each activity level, the error is similar regardless of the presence of music, because, after filtering out any unwanted signals, the algorithm handles the two cases in the same way. But with higher levels of activities, the error tends to increase. We investigated these data traces manually and found that the reception of the audio signals were very poor in those cases. This happened due to the loose contact of the earphone after a long period of jogging and continued until the user fit it well again. Overall, we observe an average error of 7.5 BPM when compared to the pulse oximeter.

Since there might be some differences between pulses obtained from the ear and from the finger tip, it is desirable to see the correlation between the two besides comparing their absolute values. Furthermore, for our application and in many medical applications, the rate of change of heart rate is of more importance than the absolute value. Figure 12(b), Figure 12(c), and Figure 12(d) show the correlations at different activity levels. The correlation coefficients are 0.85, 0.84, and 0.75 respectively. At low and medium levels, the readings are very close to the ideal (i.e. the diagonal line). But we see some non correlated horizontal and vertical points during the high activity level. The horizontal points correspond to the looseness of the earphone, and the vertical points corresponds to the inability of the pulse oximeter to report the pulse during high movements, as the device is recommended to be used at rest for its best performance.

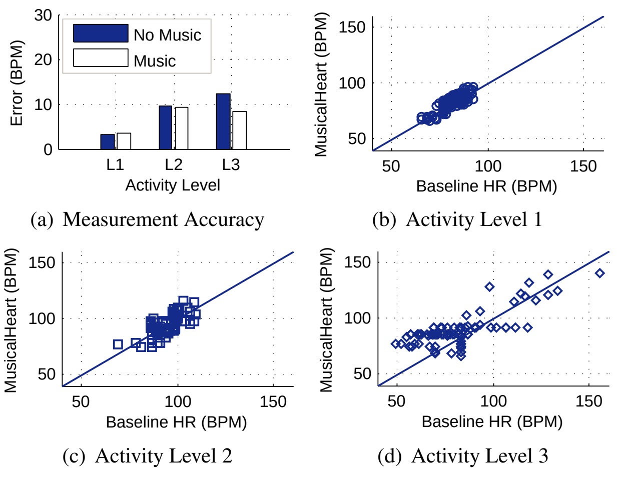


Figure 12. Performance of heart rate measurement algorithm on empirical data.

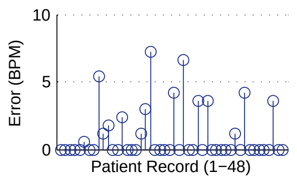


Figure 13. Performance of heart rate measurement on MIT-BIH Arrythmia dataset.

8.4.2 Performance on MIT-BIH Arrythmia Dataset

As we do not know the actual heart condition of our participants who possibly have sound health with good rhythmic heart beats, we wanted to test our algorithm on the data that are collected from actual patients. The MIT-BIH Arrythmia dataset is a standard dataset that has been used in Biomedical research community for decades. It contains raw ECG data from 48 actual patients who have known heart diseases – annotated with various information such as the heart rate. We use only the data that is obtained from ECG Lead II, since the waveform of this lead closely resembles the recording obtained from the ear. We slightly modify our algorithm to detect heart beats from this ECG data and Figure 13 shows the BPM errors for all 48 patient records. Our algorithm's average error is about 1.03 BPMs, and has zero errors for most records. In some datasets we observe error as high as 7 beats. But we manually checked that these records contain so wiggly waveforms that even human eyes cannot detect heart beats in it due to the noise.

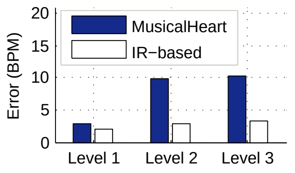


Figure 14. Comparison of heart rate measurement with an IR-based method.

8.4.3 Comparison with IR-based Technique

In this experiment, we compare the errors of our acoustic-based solution with that of an IR-based one. To do so, we design and implement a pair of Septimu-like earbuds that includes IR sensors. A microcontroller samples the IR-sensors at 50 Hz and transmits the data to a PC using the Bluetooth. We analyze the frequency spectrum of the received data and identify the peaks which correspond to the heart beats. We use a pulse-oximeter to measure the ground truth. Figure 14 compares the errors of the acoustic-based technique with that of the IR-based technique at three activity levels. For each level, we take 32 heart rate measurements and compute the mean error. We observe that, the IR-based solution beats the acoustic solution in all three levels. The mean error across all levels of activity for the acoustic sensor is 7.5 BPM and for the IR sensor, it is 2.69 BPM. Hence, this plot depicts the trade-off between the acoustic solution's multiple usages and IR-based solution's accuracy. For apps where this difference is important we would include an IR sensor in the ear bud. We leave it as a future work to combine the two sensing methods to achieve the best of both.

8.5 Evaluation of Activity Level Detection

We evaluate the accuracy of activity level detection algorithm, in which, we use only the Septimu's accelerometer data. A total of 17 users participate in this experiment. We conduct two rounds of experiments with them. First, each of the users perform exactly 3 activities, corresponding to 3 activity levels, from the list in Table 2. Each of these activities lasts about $1-5$ minutes. For each person, we separately train one classifier with 60% of the collected data, and run cross validation on the remaining 40% to obtain the confusion matrix of Table 6. We see that, the average accuracy of the learner is almost perfect at all three levels, except for a few cases where it gets confused by some L_3 activities. This has happened since some users tend to slow down after jogging for a while, which is classified by the algorithm as walking. Overall, the accuracy is 99.1%.

Actual \ Predicted	L_1	L_2	L_3
L_1	0.9998	0.0002	0
L_2	0	0.9997	0.0003
L_3	0	0.0280	0.9720

Table 6. Single activity.

Actual \ Predicted	L_1	L_2	L_3
L_1	0.989	0.011	0
L_2	0	0.951	0.049
L_3	0	0.037	0.963

Table 7. Activity sequence.

In the next round of experiments, the users perform a series of activities in sequence. The sequence, $\{L_1, L_2, L_3, L_1, L_3, L_2, L_1\}$, is chosen so that all six transitions between any two activity levels happen. Each of the activities in the sequence is 60 seconds long. After the experiment, all the collected data are classified using the same classifier

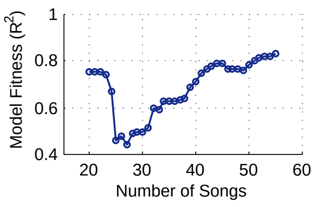


Figure 15. The fitness of model gets better as the user listens to more and more songs.

from the first set of experiments. This is done since in MusicalHeart, we first train the system for each activity level, and then the trained classifier is used to recognize different activities. The confusion matrix of this second experiment is shown in Table 7. We see some confusion between $\{L_1, L_2\}$ and $\{L_2, L_3\}$ during the transitions. But this is transient and does not affect the music recommendation algorithm. Overall, the accuracy is about 96.8%.

8.6 Fitness of System Model

Modeling human physiological response with a first order system may seem overly simplistic, but our empirical study reveals that the goodness of fit of the model is pretty accurate when the number of songs listened to by a user is 40 or more. Figure 15 plots the fitness of the model against the number of songs. In our empirical dataset, the number of songs any person has listened to is in between $20-55$. Using the history of responses from 37 people, we compute the model parameters A and B for each person. We measure the goodness of the model by computing $R^2 = 1 - \frac{var(y-\hat{y})}{var(y)}$, where y and $\hat{y}$ are the actual and predicted responses of the system. The closer the value is to 1, the better is the model.

9 Real Deployment

We deploy MusicalHeart in two real world scenarios– similar to the use cases that we described earlier in the paper. Four volunteers participate in each of the experiments. Two of them are male and two are female. All four participants are healthy and they belong to the age group of $21-30$. The participants are instructed to feel the music and to stay in rhythm of the music.

Time	Intensity	Pace
5 min	60%-70%	Walk at a comfortable pace to warm up.
3 min	70%-80%	Increase speed a few increments until working harder than the warm up pace. This is the baseline.
2 min	80%-90%	Increase speed again until working slightly harder.
3 min	70%-80%	Decrease speed back to baseline.
2 min	80%-90%	Increase speed once again until working slightly harder than baseline.
5 min	60%-70%	Decrease speed back to a comfortable level.

Table 8. The cardio exercise program that our participants take part in during the experiment.

9.1 Goal Directed Aerobics

The goal of this experiment is to demonstrate that MusicalHeart measures heart rate, detects activity levels, and suggests appropriate songs while a person is exercising. The 20 min long cardio program that our participants take part in, is described in Table 8. The program mentions the duration, target intensity, and the required change in pace. Prior to the experiment, we measure the resting heart rate and estimate the maximum heart rate of each participant – which are used to calculate the target heart rate using the equation in Section 3.3. The controller parameters at each of the 3 activity levels of a participant are estimated from the history of at least 40 previous responses of that person. This information is used by the controller to suggest appropriate music in run-time.

Figure 16 shows the variations in heart rate (normalized to intensity) during jogging for all 4 participants. The stairs represent the desired intensity, the curved lines represent the achieved intensity, and the arrows represent the activation of the control input. An upward (downward) arrow denotes a positive (negative) control signal corresponding to a suggested song that helps to rise (fall) the current heart rate. The value of control signal, u is used as the search-key in the music database to find a matching song.

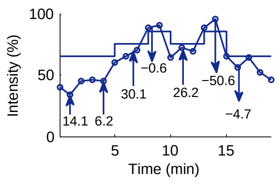
Figure 17 shows the variations in accelerometer readings corresponding to Figure 16. The curved lines represent the standard deviations of accelerometer readings. The detected activity level of a person is computed from these readings using the person specific thresholds as described in Section 6.1. The stairs represent the boundary between the detected activity levels L_2 and L_3. For example, according to the cardio program in Table 8, all 4 participant should be in L_2 for the first 5 min, then in L_3 for the next 10 min, and finally in L_2 during the last 5 min. The detected activity levels accurately match the desired levels for all 4 participants, except for the first male person (Male 1) who seems to have slowed down a little at some point during his $7-8$ min interval.

We illustrate the first control signal activation event of the first male participant as an example. The person is at the intensity level of 38%, while his desired level is 65%. From the knowledge of his resting and maximum heart rates, 70 and 170 BPM, these two intensity levels correspond to 108 and 135 BPM, respectively. Figure 17(a) shows that he is at activity level L_2 at that moment. Using the control parameters at activity level L_2 of this person, $\alpha_1 = -0.0196$, $\alpha_2 = -69.8587$, $\alpha_3 = 0.0213$, $A = 0.92$ *and* $B = 1.13$, we obtain, $u = (135 - 1.13 \times 108)/0.92 = 14.1$. The database of music is then searched for an appropriate music that has the composite feature value of 14.1. Using this control approach, overall, the mean deviation in intensity levels for the 4 participants are: 11.4%, 13.2%, 12.1%, and 11.8%, respectively.

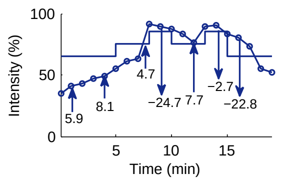
9.2 Music Recommendation via Biofeedback and Collaboration

The goal of this experiment is to demonstrate MusicalHeart's bio-feedback based rating and collaborative recommendation features. We perform a 3-day long experiment with 4 users, in which, each user uses MusicalHeart app while riding on a vehicle (e.g., bus or car) and looks for heart soothing songs. The server initially has the history of at least 40 responses from the first two users, while the other two users are new. We expect that the more a user uses MusicalHeart, the better is the quality of music recommendation.

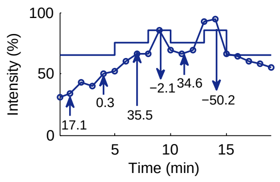
Figure 18(a) shows the automated ratings obtained from 4 users over a 3-day period. The first two users are regular users of MusicalHeart and the system therefore consistently



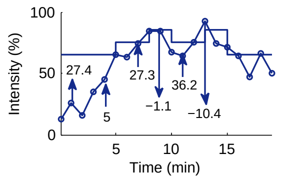
(a) Male 1



(b) Female 1

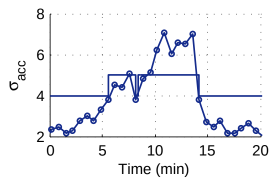


(c) Male 2

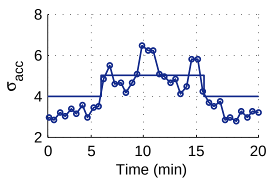


(d) Female 2

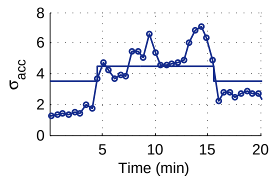
Figure 16. The desired intensity, achieved intensity and activation of control signals are shown.



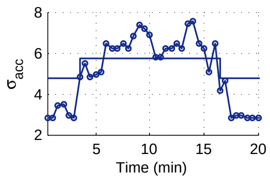
(a) Male 1



(b) Female 1

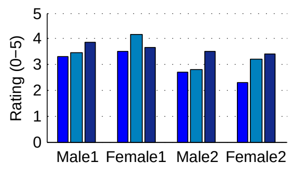


(c) Male 2

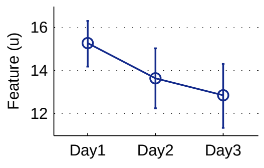


(d) Female 2

Figure 17. Standard deviation of accelerometer data and boundary between activity levels L_2 and L_3 are shown.



(a)



(b)

Figure 18. (a) The automated ratings for the new users are increased. (b) The reduction of u indicates increased accuracy of soothing music suggestion.

suggests high rated songs to them with average ratings of 3.56 and 3.78. For the new users of the system, initially, the quality of suggestion is in the range of $2.32-2.79$, but the more they use the app, the better the quality of suggestion gets, i.e., $3.42-3.52$. The improvement happens due to the learning and collaborative nature of the recommendation algorithm. Figure 18(b) shows the value of the composite feature, u of the suggested songs on each day. The composite feature being highly correlated to raising the heart rate, the less its value, the better it is in calming down the heart. We see that the average value of u gets lower on each day, which indicates that MusicalHeart is indeed suggesting heart soothing song to its users. Note that, the values of u in this experiment are not comparable to those in the previous experiment since all 4 users are in activity level L_1 which is different.

10 Related Work

Music has been shown to improve sleep quality [19, 25], cope with stress [24], improve performance [35, 14], influence brain activity [21], and increase motor coordination [8]. However, we only consider the response of the heart to music. Studies have shown that exercise intensity and tempo are correlated linearly [22], and relaxing music (e.g., Bach, Vivaldi, Mozart) result in reduction of heart rate and its variability [15]. We perform a similar study, but the difference is ours involves smartphone users who are mobile and whose activity levels change.

The smartphones apps that measure heart rate use 3 basic principles: camera, microphone, and accelerometer based. Camera based apps (Instant Heart Rate, PulsePhone) detect pulses by measuring the changes in the intensity of light passing through the finger. Microphone based apps (Heart Monitor, iRunXtream) require the user to hold the microphone directly over heart, or neck, or wrist to detect beats. Accelerometer bases techniques (iHeart) are indirect, in which, the user measure his pulse with one hand (e.g. from neck) and taps or shakes the phone with other hand in the rhythm of the heart beats. Our approach in MusicalHeart is microphone based, but the difference is, in all of the existing apps, the heart rate monitoring requires active engagement of the user, whereas MusicalHeart obtain heart beats from the ear without interrupting the user's music listening activity. Other approaches that detect heart beats from the ear [33, 10] use a combination of infrared LED and accelerometers.

Several works are related to our activity detection algorithm. [34] concludes that it is practical to attribute various activities into different categories which is similar to our activity levels. [36] describes an accelerometer based activity classification algorithm for determining whether or not the user is riding in a vehicle, or in a bus, or another vehicle for cooperative transit tracking. In our work, we only require to detect whether someone is indoors vs. outdoors, and his current speed. [11] presents a mobile sensing platform to recognize various activities including walking, running and cycling. [28] presents a wearable platform for motion analysis of patients being treated for neuromotor disorders. [13] detects user's caloric expenditure via sensor data from a mobile phone worn on the hip. But these works require the user to wear sensors at specific positions of the body. In our case, the user naturally wears the earphone on the ear. [7] performs activity recognition from user-annotated acceleration data, which is different than our unsupervised learning. [6] presents a high performance wireless platform to capture human body motion. In our work, we do not require such a high speed communication to the server.

Existing approaches for automatic music recommendation are: using one or more seed songs [32, 16], retrieve similar music by matching the tempo [37], and learning the habit (e.g., music genre, preferences, or time of day) of the user [27, 5]. None of these consider any physiological effects of music on the user. Physiological information, such as gait and heart rate, have been considered in [30, 12, 29, 9].

But the downside of these approaches are, first, the user has to use a separate device (e.g. ECG or pulse oximeter) which is an inconvenience, second, the music recommendation is based on an overly simplistic and general rule of thumbs, i.e., to suggest music with a matching tempo, which is not personalized, and third, the recommendation is not situation aware and mostly targeted to a single scenario of jogging.

11 Conclusion

In this paper, we introduce Septimu, a wearable sensing platform consisting of a pair of sensor equipped earphones, and as an application of it, we design and implement, MusicalHeart which is a novel bio-feedback based, context aware, automated music recommendation system for smartphones. We provide empirical evidence that the system is capable of detecting heart rate from the ear which is $75\% - 85\%$ correlated to the ground truth and has an average error of 7.5 BPM. The accuracy of activity level inference of the system is on average 96.8%. We deploy MusicalHeart in two real world scenarios and show that it helps the user in achieving a desired exercising intensity with an average error of less than 12.2%, and its quality of music recommendation improves over time.

Acknowledgments

This work was supported, in part, by NSF grant EECS 1035771.

12 References

[1] Fingertip pulse oximeter. `http://www.naturespiritproduct.com`.

[2] Handheld ecg monitor (md100b). `http://www.choicemmed.com/`.

[3] Power meter. `http://msoon.com/LabEquipment/PowerMonitor/`.

[4] F. J. Aherne, N. A. Thacker, and P. Rockett. The bhattacharyya metric as an absolute similarity measure for frequency coded data. *Kybernetika*, 34(4):363–368, 1998.

[5] A. Andric and G. Haus. Automatic playlist generation based on tracking user's listening habits. *Multimedia Tools and Applications*, 29(2):127–151, June 2006.

[6] R. Aylward and J. A. Paradiso. A compact, high-speed, wearable sensor network for biomotion capture and interactive media. In *6th International Conference on Information Processing in Sensor Networks 2007 (IPSN '07)*, pages 380–389, Cambridge, MA.

[7] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. *IEEE Pervasive Computing*, 3001:1–17, 2004.

[8] G. Bernatzky et. al. Stimulating music increases motor coordination in patients afflicted with morbus parkinson. *Neuroscience Letters*, 361(1–3):4–8, May 2004.

[9] J. T. Biehl, P. D. Adamczyk, and B. P. Bailey. Djogger: a mobile dynamic music device. In *ACM SIGCHI Conference on Human Factors in computing systems 2006 (CHI '06)*, Montreal, Canada.

[10] P. Celka, C. Verjus, R. Vetter, P. Renevey, and V. Neuman. Motion resistant earphone located infrared based heart rate measurement device. *Biomedical Engineering*, Feb 2004.

[11] T. Choudhury et. al. The mobile sensing platform: An embedded activity recognition system. *IEEE Pervasive Computing*, 7(2):32–41, April 2008.

[12] R. de Oliveira and N. Oliver. Triplebeat: enhancing exercise performance with persuasion. In *10th International Conference on Human Computer Interaction with Mobile Devices and Services 2008 (MobileHCI '08)*, pages 255–264.

[13] T. Denning, A. Andrew, R. Chaudhri, C. Hartung, J. Lester, G. Borriello, and G. Duncan. Balance: towards a usable pervasive wellness application with accurate activity inference. In *10th Workshop on Mobile Computing Systems and Applications 2009 (HotMobile '09)*.

[14] J. Edworthy and H. Waring. The effects of music tempo and loudness level on treadmill exercise. *Ergonomics*, 49(15):1597–610, Dec 2006.

[15] J. Escher and D. Evequoz. Music and heart rate variability. study of the effect of music on heart rate variability in healthy adolescents. *Praxis*, 88(21):951–2, May 1999.

[16] A. Flexer, D. Schnitzer, M. Gasser, and G. Widmer. Playlist generation using start and end songs. In *9th International Conference on Music Information Retrieval 2008 (ISMIR '08)*, pages 173–178.

[17] G. Friesen, T. Jannett, M. Jadallah, S. Yates, S. Quint, and H. Nagle. A comparison of the noise sensitivity of nine qrs detection algorithms. *Biomedical Engineering*, 37(1):85 –98, Jan 1990.

[18] A. Goldberger, L. Amaral, L. Glass, J. Hausdorff, P. Ivanov, R. Mark, J. Mietus, G. Moody, C.-K. Peng, and H. Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):215–220, Jun 2000.

[19] L. Harmat, J. Takacs, and R. Bodizs. Music improves sleep quality in students. *Advanced Nursing*, 62:327–335, May 2008.

[20] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback Control of Computing Systems*. Wiley-IEEE Press, Aug 2004.

[21] N. Jausovec, K. Jausovec, and I. Gerlic. The influence of mozart's music on brain activity in the process of learning. *Clinical Neurophysiology*, 117(12):2703–14, Dec 2006.

[22] C. I. Karageorghis, L. Jones, and D. C. Low. Relationship between exercise heart rate and music tempo preference. *Research Quarterly for Exercise and Sport*, 26:240–250, 2006.

[23] Y.-S. Kuo, S. Verma, T. Schmid, and P. Dutta. Hijacking power and bandwidth from the mobile phone's audio interface. In *1st ACM Symposium on Computing for Development 2010 (DEV '10)*.

[24] E. Labbe, N. Schmidt, J. Babin, and M. Pharr. Coping with stress: the effectiveness of different types of music. *Applied Psychophysiology Biofeedback*, 32(3–4):163–8, Dec 2007.

[25] H. Lai and M. Good. Music improves sleep quality in older adults. *Advanced Nursing*, 49(3):234–44, Feb 2005.

[26] O. Lartillot. Mirtoolbox user's manual. *Finnish Centre of Excelence in Interdisciplinary Music Research*, Dec 2011.

[27] N. H. Liu, S. J. Hsieh, and C. F. Tsai. An intelligent music playlist generator based on the time parameter with artificial neural networks. *Expert Systems with Applications*, 37(4):2815–2825, April 2010.

[28] K. Lorincz, B.-r. Chen, G. W. Challen, A. R. Chowdhury, S. Patel, P. Bonato, and M. Welsh. Mercury: a wearable sensor network platform for high-fidelity motion analysis. In *7th ACM Conference on Embedded Networked Sensor Systems 2009 (SenSys '09)*, pages 183–196, Berkeley, CA.

[29] N. Oliver and F. Flores-Mangas. Mptrain: a mobile, music and physiology-based personal trainer. In *8th International Conference on Human Computer Interaction with Mobile Devices and Services 2006 (MobileHCI '06)*, pages 21–28, Helsinki, Finland.

[30] N. Oliver and L. Kreger-Stickles. Papa: Physiology and purpose-aware automatic playlist generation. In *7th International Conference on Music Information Retrieval 2006 (ISMIR '06)*, pages 250–253.

[31] J. Pan and W. Tompkins. A real-time qrs detection algorithm. *Biomedical Engineering*, 32(3):230–236, March 1985.

[32] J. C. Platt, C. J. Burges, S. Swenson, C. Weare, and A. Zheng. Learning a gaussian process prior for automatically generating music playlists. In *Advances in Neural Information Processing Systems 2001 (NIPS '01)*, pages 1425–1432.

[33] M.-Z. Poh, K. Kim, A. Goessling, N. Swenson, and R. Picard. Heartphones: Sensor earphones and mobile application for non-obtrusive health monitoring. In *International Symposium on Wearable Computers 2009 (ISWC '09)*, pages 153–154.

[34] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. In *17th Conference on Innovative Applications of Artificial Intelligence 2005 (IAAI '05)*, pages 1541–1546, Pittsburgh, PA.

[35] S. Simpson and C. Karageorghis. The effects of synchronous music on 400-m sprint performance. *Sports Sciences*, 24(10):1095–102, Oct. 2006.

[36] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson. Cooperative transit tracking using smart-phones. In *8th ACM Conference on Embedded Networked Sensor Systems 2010 (SenSys '10)*, pages 85–98, Zurich, Switzerland.

[37] X. Zhu, Y.-Y. Shi, H.-G. Kim, and K.-W. Eom. An integrated music recommendation system. *Consumer Electronics*, 52(3):917–925, Aug. 2006.