Pulse — Dynamic & Interactive Music Visualization using Real-Time Heartbeat Data

Dhairye Kamra
drk2399@rit.edu
Dept. of Computer Science
Rochester Institute of Technology

Abstract—An individual’s heartbeat can be verbose about a plethora of characteristics including, but not limited to: a heart disease/attack, excitement, anxiety, reaction, exercise or even romance. These beats and factors contribute towards drawing a vivid picture of a person’s way of living. Complex heartbeat measuring algorithms and devices have been designed to make use of this vital information and map it to a general perspective in order to elaborate and digitize critical observations, which were not explicitly known or given before. In our approach, we leverage the aforementioned technique and the characteristics of the human heartbeat to dynamically visualize human heartbeat using real-time pulse data, reacting to the instantaneous live music setting, while using cheap portable sensors paired with common everyday mobile devices.

I. INTRODUCTION

Traditionally, monitoring a user’s heartbeat was only for medical or conservative purposes. As time advanced, these devices became more portable using the advances of the powerful smart phone camera and flash light to detect the reflection of the pulse from the finger, but, such an application requires the operation of a high powered flash [1], and a painstaking amount of time. In a live concert music setting, this would be highly undesirable given the critical features of a system.

Leveraging technology, it has become much easier to make heart rate monitoring devices extremely small, efficient and cheap. Also, having a virtually tangible form of music makes it easier to bring the audience closer to the performer while giving them the ability to gain instantaneous feedback [1]. It also provides the performer with the audience statistics for each segment of the concert. In this paper, we introduce such a technology, designing a heartbeat driven music feedback system using a portable and modular infrastructure.

The sections in the paper are arranged as follows: Section 2 talks about the System Architecture, namely, the Arduino board, pulse sensor, Android application, Java server and very briefly about the visualization scheme. Section 3 talks about the actual working of the project with sub-points including the Generic Controller, virtual pin code authentication, Accelerometer-Biased accuracy and the low-pass filter implementation. Section 4 talks about the performance and energy consumption with different data transfer rates with respect to the mobile application and the pulse sensor. Section 5 talks about the related work in this space and section 6 concludes the paper.

II. SYSTEM ARCHITECTURE

The proposed pulse system consists of several modules in the hierarchy, but the only extra overhead devices are the light blue bean (Arduino Board) and the pulse sensor, which are connected together to measure the human heartbeat. We wrap these small devices into one single attached unit which can be easily tapped and attached to the finger. The entire connection on the client side (audience) are wireless, that is, the communication from the pulse sensor to the android device is done over Bluetooth 4.0 (Low Energy) and the information exchange of the Android phone with the server is done over WiFi. Figure 1 depicts the system architecture flow between the different modules in the system. The following subsection gives a detailed elaboration & understanding of each of these modules.

A. Light Blue Bean & Pulse Sensor

The light blue bean is a tiny (about 3 inches) board equipped with a programmable Arduino chip [3], where the chip is capable of intercepting data from multiple sensors, and enabling the right actuators/procedures in the environment. In our case, the pulse sensor is the heart-rate sensing device which is attached physically to the bean. The pulse sensor is a tiny light emitting device, which imparts high power green LED light that reflects some light back on the detection of the pulse (blood passing in the finger) in the finger[2]. The device features an ambient light sensor to cancel out the noise based on the environment and an accelerometer to detect movements in the three dimensional space. We will be talking about the exploitation of this accelerometer feature to cancel out some noise from the pulse sensor due to excessive hand movements.

B. Android Application

Another module of our system is the android application, which primarily absorbs the readings from the bean and sends
this data to the server. The application consists of three basic stages:

1) Connection: This phase validates the connection between the application and the server, if true, it checks the persistence of the user’s existence in the database. If the user is already registered, the app prompts him to wait until the event starts. This is done to protect the sensor and the phone battery levels. If not, the user is prompted to Register the device.

2) Registration: This step prompts the user to fill out his details, like the name and the seat number (used for data collection and analysis). Along with this, the user also scans the QR code to personalize the sensor, which would exclusively be paired to that single mobile device each time. This is done by mapping the unique IDs of the sensor to that of the device. This combination also creates a unique hash and sends it to the sensor for all future authentications. Once this is completed, the above step is repeated.

3) Data communication: On the receipt of an event being started, the GUI of the application moves on to the pulse page, where it starts discovering its personal pulse sensor and starts accepting heartbeat readings visible to the user. These readings from the sensor are sent to the server for persistence and analytics. The communication of the RMQ with the server also consists of control messages explained in the sections to follow.

The application holds several other modules to handle the communications and the control messages which will be explained later in the paper.

C. Server & Backend

The server acts as a central authority of managing the devices, events and the direct interaction with the database. The server also maintains a connection to the lightweight RabbitMQ queues and sends the subscriber-related messages to these queues. It is also responsible to create a new event with attributes having the event id, name, location and the current date.

The server also behaves as the data controller, which can alter the data flow rates (from the phone or even the sensor) while also applying various filters to fetch the data, only from the specific devices based on the problem characteristics. These messages are sent across the RMQ structure, while controlling different aspects, such as receiving data based on user location, i.e., seat numbers or the random color they are assigned during the registration process.

The back-end is implemented using the simple, structured and lightweight MySQL database consisting of four tables to persist different data about the user and heartbeat info. This database is used by the visualization module to take the data in dumps and present it using different descriptive schemes signifying the user heartbeat flagged with the registered name.

D. Visualization Scheme

The visualization scheme sits separate from the aforementioned setup. It is enabled by a separate NodeJS server fetching the data from the database. The query picks up the most recent data as soon as it becomes available and portrays it on the canvas. The schemes range from a simple line acceleration giving the heartbeats of the user in space, to a more complex...
sphere structure keeping fast movements in the center and moving outwards. It mainly shows the user feedback towards the live concert setting.

III. IMPLEMENTATION, WORKING & ENHANCEMENTS

The overall interaction of the full system is depicted in the timeline diagram depicted in figure 1. To gain a more granular insight to what the server and application are trying to accomplish is explained below.

A. Generic controller

During the time of a live concert, it would presumably be tough to tweak changes and control the working of the devices and sensor itself based on the instantaneous nature of the song. To have that kind of granular control, i.e., to isolate and manipulate the working of each device based on the music segment was needed, by having some central controller having the entire view of the system like the devices subscribed to a particular event, seating locations, color, name, etc. The central controller is able to manipulate the following using a notification message:

- Rate Flow Wi-Fi ($R_W$): The controller can signal the devices subscribed to an event, to slow down or step up the transfer rate based on the nature of the music segment.
- Rate Flow Bluetooth ($R_B$): Similarly, to control the sending rate of the pulse sensor, transferring over Bluetooth low-energy, in order to enhance the power consumption even further. It is observed that the sensor device can last up to 4 hours as opposed to the default approach which lasted only about 3 hours on full use.
- Device Control: The controller can also suspend or start receiving from different queues which could be related to random colors or seat positions according to the map leveraging the RMQ structure.
- Create/Start/Stop Events: The controller, like conventionally done, can register about an event with all the details mentioned above, and also able to notify the devices about the starting/stopping of that event.

B. Virtual Pin-Code

The proposed system deals with highly personal heartbeat data of an individual. It can be seen as a matter of grave concern for the pulse sensors to be strictly paired with the corresponding device as another device must not collect data on a different ID. To accomplish this, we used a hash computed by the android device to be stored on the pulse sensor itself. This hash can be seen as follows:

$$H(MDevice_{\text{Hex}} + \text{PDevice}_{\text{Pair}})$$

The $MDevice_{\text{Hex}}$ is a part of the unique mobile ID and $\text{PDevice}_{\text{Pair}}$ is the bean ID obtained after physically scanning the QR code by the device camera. The first time a device tries to make a connection to a new bean, the value of this hash is stored on the bean. For each subsequent connection attempt, this hash must match for the pairing device to successfully connect to the sensor.

C. Sensor accuracy

One of the prime modules of the system is to measure the heartbeat. The pulse sensor is textbf{not} really the state-of-the-art system, as it comprises of only one light based sensor to measure the signals (as opposed to 4 sensors in sophisticated devices like the apple watch). Due to this, the sensor is very susceptible to environmental noise and movements. In order to minimize these noises and gain a highly accurate reading of the heartbeat, other contributors worked on implementing the following two mechanisms:

1) Accelerometer Biased Accuracy: As mentioned above, each LBB is equipped with an accelerometer. We leverage this feature to calculate a threshold value which gives the permissible movement across the x, y and z axes. If the movement is greater, the reading is discarded. This is especially done to cancel out all the movements based on arbitrary hand movements during the concert.

D. Low-Pass Filter Implementation

The heartbeat can have irregularities, but some are introduced because of wireless or environmental noise. Also, our implementation deals with musical feedback and not intended for medical purposes. Given this, we can easily deduce that the heartbeat will not fluctuate arbitrarily to a non-believable value. The LPF will be able to cancel out such anomalies present in the bucketed readings.

IV. DATA COLLECTION, PERFORMANCE AND ENERGY CONSUMPTION ANALYSIS

In this section we focus on discussing about the data collection results. Every application is primarily judged on its utility versus the power it consumes to provide that functionality. We try to reason out the practicality of the data rate and other factors to the justified power consumption.

A. Experimental Setup

The setup consisted of 6 Android devices along with their personal sensors successfully connected on the network. To gain granular accuracy corresponding to real application usage, the devices were running with minimum number of required background services and all other applications shut off. We used the popular ODroid power measurement device, which functions by removing the device battery and connecting the power outlets of the devide to that of the ODroid. We then measured the energy consumption of the application by tweaking the data rate flow.

B. Sending rate Vs. Power

As we studied above, the rate of data flow is manipulated by the server. The number of data points (BPM) along with the song segment gives the crucial information about the user reaction to the particular style/pitch/tempo/sentiment of the song itself. But it is impractical to keep the same data flow and consume more energy for the entire length of the concert, as the parts of the song may be redundant or move with the same pace for a long segment before another rise or cliff. Therefore,
it is within good reason to handle the data rate based on the
critical nature of the music segment itself. Figure 3 compares
the power consumption with different phases of the device. It
is very evident that the rate of sending has a reasonable impact
on the battery life of the device as it makes significant use
of the Wi-Fi channel.

Looking at the figure, we can deduce that it is fairly
important to regulate the data flow between different segments
of the song to conserve the battery life for a truly energy
efficient application.

Similarly, the sending rate over the Bluetooth channel, i.e.,
the sending rate from the sensor to the device can be optimized
by controlling the consumption of the channel by the light blue
bean.

C. Correlation and Construction of the Raw Waveform

To find the correlation between the music and the heartbeat,
it was important to construct the waveform for the raw signal
data and observe the subtle changes between the peaks, cliffs
and the diacritic notch under different time boxed windows.
The pulse sensor throws out an amplitude signal every 2
milliseconds, which was too fast for the serial log or any en-
queuing mechanism to cope up with, and hence we had to
slow it down by 4 milliseconds to collect a signal every 6ms.
Figure 2 depicts this raw constructed baseline waveform of
one of the subjects. The waveform shows many broken parts
circled in red due to the lag introduced between the collected
and the original point. Also, it is observably difficult to overlap
the non-baseline heartbeats due to the difficulty in attaining
a similar baseline pattern due to the broken ridges. Also, it
can be said that, due to the lightweight processing power of
the bean, it was difficult to strain all the points needed to
reconstruct the waveform, and hence making it difficult to spot
the granular changes in the baseline heartbeat.

To provide this correlation on a more coarse grained scale,
we used the processed signal, i.e., the BPM itself to find some
relevance with the music. Figure 5 shows the relationship
between the baseline heartbeat, heartbeat while listening to
the music, for one male subject averaged over 5 eight minute
readings.

This can be compared to the visualized structure of the
music form itself, which is depicted in figure 4 above. The
observations shows correlation between the high and low
peaks of the song, with the BPM ranging from 72 to 128.
Also, the music file consisted a mix of varying song types
with different pitches, tempo, genre and aggression. It is also
important to point out that BPM may not be the best technique
to judge this correlation as the signal is already processed and
thereby the pulse sensor must be powered by something more
powerful than a LBB.

V. CONCLUSION

In this paper, we propose a visualization system using the
dynamic nature of the human heartbeat, which is responsive to
many environmental or natural changes, including music. The
system is able to instantaneously collect the heartbeat data and present it on various different patterns.

To enhance the implementation, we develop a smart controller which is able to manipulate flows, data acquisition and the events of the entire hierarchy from one single place. To implement security, we implement a virtual pin code, capable of authenticating a device without physically having to pair in the Bluetooth settings. We also revamped the entire connection scheme, while providing more accurate heartbeat readings leveraging the accelerometer and a comprehensive LPF. We were able to correlate the human heartbeat with the music using a common file applied tested on a variety of test subjects averaging the data across multiple attempts, while also being able to deduce a comprehensive power consumption for the android device and the pulse sensor with different rate flows.

REFERENCES


