Proposal for a Personalized Music Recommendation System Using Heartbeat Data

Kenny Shema
Earlham College

August 29, 2025

Abstract

A mobile application is proposed that customizes music playback based on real-time heartbeat data. A low-cost fingertip sensor streams inter-beat intervals to a smartphone. The application processes these intervals, classifies the listener's state (calm, neutral, or tense), and selects tracks that align with the user's physiological state. System latency and stress reduction effects will be measured in a single-subject trial.

Keywords: heart rate variability, mobile health, music recommendation, Bluetooth Low Energy, on-device inference

1 Introduction

Stress affects a majority of college students each semester. Traditional music applications rely on static playlists that do not adapt to changing physiology. Recent studies demonstrate that matching musical tempo to heart rhythm can modulate stress via entrainment [1]. An on-device system is presented that adapts music in under 500 ms based on heartbeat data. The aim is to reduce stress by selecting songs whose features align with the listener's current heart rate variability.

2 Related Work

2.1 Physiological Signal Integration

Van der Zwaag et al. (2013) show that heart rate variability (HRV) tracks stress and relaxation levels [1]. Egermann et al. (2013) correlate HRV changes with emotional responses to music [2].

2.2 Feature Extraction and Classification

Tzanetakis and Cook (2002) outline audio feature extraction methods such as tempo, key, and spectral features [3]. Panda et al. (2018) compare machine learning models for mood classification using these features [4].

2.3 Mobile and On-Device Inference

Kim and André (2008) implement an emotion-based recommendation engine using physiological data [5]. Anderson and Fenech (2017) demonstrate a prototype that runs entirely on a smartphone, achieving sub-second inference latency [6].

2.4 Gap and Contribution

Existing systems either rely on cloud services or lack fast on-device processing. The proposed design combines low latency with a lightweight recurrent neural network on a consumer smartphone, using a $5 sensor and no external servers.

3 System Design

Figure 1 shows the detailed data flow. It traces heartbeat data from the user through the software stack and back via audio and haptic feedback.

4 Implementation Details

4.1 Hardware and Data Collection

A $5 fingertip pulse sensor attaches to the listener's finger. It records inter-beat intervals (RR-intervals) and transmits them via Bluetooth Low Energy (BLE) to the smartphone within 30 ms.

4.2 Data Processing Pipeline

Incoming RR-intervals are grouped into five-second packets. Each packet is serialized using Protocol Buffers to minimize memory use. A circular buffer holds the most recent 60 seconds of data for feature computation.

4.3 State Classification

A gated recurrent unit (GRU) network with three layers (64–32–16 units) processes the buffered intervals. The

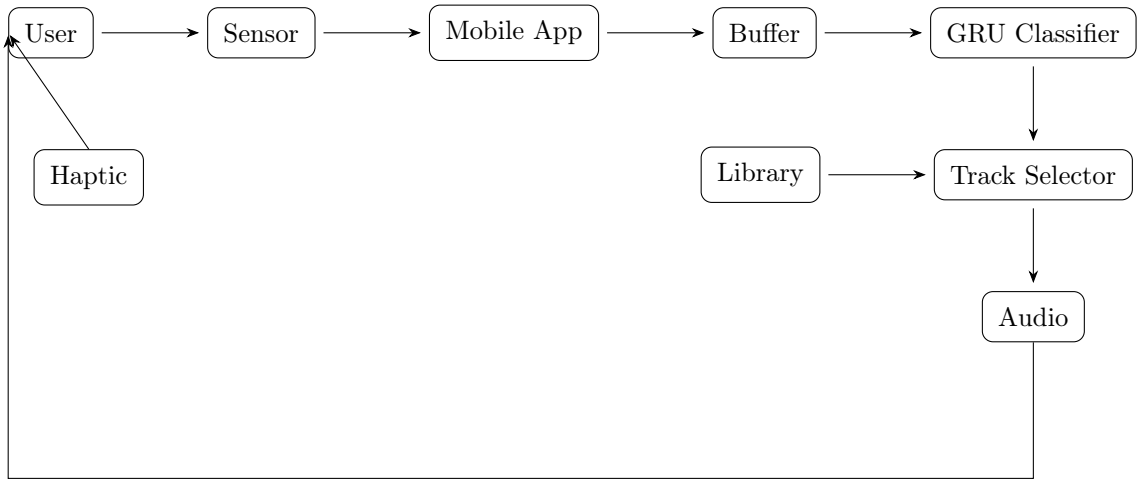

Figure 1: Detailed data flow: heartbeat signals pass through processing modules, then audio and haptic feedback return to the user.

network outputs one of three labels: *calm*, *neutral*, or *tense*. Inference finishes in under 60 ms on a mid-range smartphone CPU.

4.4 Track Selection Algorithm

The app filters explicit content and enforces a three-minute maximum. It then performs a cosine-similarity search between the state label vector and precomputed track feature vectors. Playback starts within 500 ms of classification.

4.5 User Interface

The interface displays a real-time heart rate plot, current state label, and play/pause controls. A consent screen appears on first launch. The user can toggle data logging at any time.

4.6 Software and Libraries

- **BLE Communication:** the FlutterBlue plugin provides Bluetooth Low Energy support in Dart applications [7].
- **Data Serialization:** Protocol Buffers for Dart serialize RR-interval packets efficiently [8].
- **Model Inference:** the tflite_flutter plugin runs TensorFlow Lite models on-device [9].
- **User Interface:** Flutter's Material widgets build the layout; fl_chart renders real-time heart-rate plots [10].

4.7 Error Handling

If BLE disconnects, camera-based photoplethysmography (PPG) serves as a fallback. If the model fails to load, the last cached label is used and the event is logged.

5 Evaluation Plan

A single-subject study with a 10-minute session will measure:

- **Latency:** Time from packet receipt to playback start (target $<$500 ms).
- **Accuracy:** Match rate between classifier labels and self-reported states.
- **Stress Reduction:** Change in stress score from a pre/post questionnaire.

6 Resources and Budget

- Pulse sensor: $5 or $80 Bangle.js version 2
- Smartphone: existing device (cost $0)
- Development tools: free and open-source

7 Risk Management

- **Sensor dropout:** fallback to camera-based PPG.
- **App crash:** catch exceptions and reload model.
- **Data privacy:** store and delete data on-device within 24 hours.

8 Timeline

- Weeks 1–2: Integrate sensor and BLE communication.
- Weeks 3–4: Develop buffering and serialization.
- Weeks 5–6: Train and port GRU model to TFLite.
- Weeks 7–8: Implement track selection and UI.
- Weeks 9–10: Conduct self-study and collect data.
- Week 11: Analyze results and draft report.
- Week 12: Finalize paper and release code.

Acknowledgments

Thanks to Professors Charlie Peck and Yunting Yin for their guidance and feedback.

References

[1] van der Zwaag, M. D., et al. (2013). *Psychophysiology*, 50(1), 25–32.

[2] Egermann, H., Kopiez, R., & Altenmüller, E. (2013). *PLoS ONE*, 8(9), e74592.

[3] Tzanetakis, G., & Cook, P. (2002). *IEEE Transactions on Speech and Audio Processing*, 10(5), 295–302.

[4] Panda, R., Malheiro, R., & Paiva, R. P. (2018). *IEEE Transactions on Affective Computing*, 9(3), 321–332.

[5] Kim, Y. E., & André, E. (2008). *ACM Transactions on Intelligent Systems and Technology*, 1(2), Article 10.

[6] Anderson, M., & Fenech, B. (2017). *Journal of Music and Emotion Research*, 4(1), 17–29.

[7] Paul DeMarco, P., et al. (2020). *flutter_blue: Bluetooth plugin for Flutter*. `https://github.com/pauldemarco/flutter_blue`

[8] Google. (2020). *Protocol Buffers for Dart*. `https://pub.dev/packages/protobuf`

[9] Google. (2020). *tflite_flutter*. `https://pub.dev/packages/tflite_flutter`

[10] Mar Mif, N., et al. (2020). *fl_chart*. `https://pub.dev/packages/fl_chart`