2021 43rd Annual International Conference of the
IEEE Engineering in Medicine & Biology Society (EMBC)
Oct 31 - Nov 4, 2021. Virtual Conference

RRMonitor: A Resource-Aware End-to-End System for Continuous Monitoring of Respiration Rate Using Earbuds

Tousif Ahmed[1], Md Mahbubur Rahman[1], Mohsin Yusuf Ahmed[1], Ebrahim Nemati[1], Minh Dinh[2], Nathan Folkman[2], Jilong Kuang[1], and Alex Gao[1]

***Abstract*— Respiration rate is considered as a critical vital sign, and daily monitoring of respiration rate could provide helpful information about any acute condition in the human body. While researchers have been exploring mobile devices for respiration rate monitoring, passive and continuous monitoring is still not feasible due to many usability challenges (e.g., active participation) in existing approaches. This paper presents an end-to-end system called *RRMonitor* that leverages the movement sensors from commodity earbuds to continuously monitor the respiration rate in near real-time. While developing the systems, we extensively explored some key parameters, algorithms, and approaches from existing literature that are better suited for continuous and passive respiration rate monitoring. *RRMonitor* can passively track the respiration rate with a mean absolute error as low as 1.64 cycles per minute without requiring active participation from the user.**

***Clinical relevance*— This work enables continuous monitoring of respiration rate during daily life that has significant potential for detecting abnormal changes in respiration rate.**

I. Introduction

Continuous monitoring of respiration rate has a significant impact on general health and wellness. Typically, an adult takes 12-20 breaths per minute (BPM), and any sudden changes in regular breathing rate often indicate the onset of lung condition deterioration and acute illness [1]. For example, several clinical studies suggest that an adult with a high (>24 BPM) or low respiratory rate (<8 BPM) is likely to have an underlying health problem [2], [3]. Fieselmann et al. [4] reported that a breathing rate over 27BPM in the last 72 hours could be a strong predictor of cardiac arrest. While clinical researchers emphasized the importance of monitoring the respiration rate daily, it is challenging to monitor outside clinical contexts as monitoring requires specialized devices.

In recent years, researchers have been actively investigating a more accessible approach to monitor the respiration rate by utilizing low-cost mobile [5], [6] and wearable [7], [8] devices. By leveraging the motion sensors from mobile devices, previous works showed that it is feasible to estimate the respiratory rate outside of the clinical contexts. While the proposed methods are promising, there can be many challenges in developing the algorithms on mobile devices. For example, some solutions may consume significant computing resources, making the approaches infeasible for continuous monitoring. We also need to make several trade-offs between accuracy and resource consumption to keep the monitoring solutions lightweight. We do not have a good understanding of such trade-offs that we need to make to develop an end-to-end solution for respiratory rate monitoring.

In this work, we extensively investigate the key trade-offs to develop a lightweight end-to-end system called *RRMonitor* for passive and continuous respiration rate monitoring. *RRMonitor* utilizes the motion sensors from commodity earbuds and estimates the breathing rate by employing a very lightweight zero-crossing rate-based breathing algorithm. In this work, we have focused on earbuds by considering several design decisions. First, recent work [9] has shown that it is feasible to monitor the respiration rate using the inertial sensors from earbuds with reasonable accuracy. Second, while most solutions require active participation from the user (e.g., users need to put the devices on their chest or abdomen), earbuds can provide a more passive solution without requiring too much attention from the user. More than 30% people spend roughly 3-4 hours per day listening to music on their headphones/earbuds [10]. While the users are using their earbuds for their daily purposes, *RRMonitor* can continuously monitor the breathing rate without interfering in their daily activity. Third, earbuds are very closely positioned to the nose, a crucial breathing organ, enabling a more robust solution. Finally, modern earbuds can be coupled with smartphones; hence, we can leverage the computing power of mobile devices for processing. We only focused on motion sensors instead of other high fidelity sensors (e.g., audio) to reduce computational complexity and memory usage.

We develop *RRMonitor* utilizing the motion sensors collected from Samsung earbud and smartphone. We investigate the key parameters and algorithms for *RRMonitor* on a breathing dataset collected from 30 individuals. Our system can monitor the respiration rate in near real-time without consuming significant resources (2-3% resources). The main contributions of this paper are: 1) We present an end-to-end system for breathing monitoring using the motion sensors from commercial earbuds, and 2) We empirically investigate the key parameters and algorithms that are better suited for a lightweight continuous monitoring solution.

II. Related Work

Previous works extensively studied techniques for extracting breathing rates utilizing inertial and acoustic sensors from commercial mobile and wearable devices. Nam et al. [11] use the sound recordings by placing a mobile phone underneath

[1]T. Ahmed, M. Rahman, M.Ahmed, E.Nemati, J.Kuang, and A. Gao are with Digital Health Lab, Samsung Research America, Mountain View, CA, USA (corresponding author email: tousif.a@samsung.com).

[2] M. Dinh and N. Folkman are with Samsung Design and Innovation Center, San Francisco, CA, USA.

the nose to estimate the respiration rate. While audio-based approaches have demonstrated promising results, processing audios on the phone require heavy CPU usage, which is not feasible for continuous monitoring. Although earbuds are exceptionally well-positioned for acoustic-based breathing monitoring, in this work, we only considered motion sensors to keep the system lightweight.

Approach	Device	Algorithm	MAE (BPM)
Zephyr [5]	Phone	FFT	0.6 (median)
InstantRR [6]	Phone	FFT, ZCR, & Peak	0.85-4.82
BioWatch. [7]	Watch	FFT	0.19-0.72
SleepMonitor [12]	Watch	FFT	0.72
WearBreathing [8]	Watch	RF + CNN	1.09-2.05
Röddiger et al. [9]	Earphone	FFT	2.62
BioGlass [13]	Glass	FFT	0.8-1.77

TABLE I: Overview of approaches that use inertial sensors to estimate breathing rate (MAE = mean absolute error).

Table I presents an overview of the approaches that leverage the inertial sensors from the phone, watch, and head-mounted devices to estimate respiration rate. As can be seen in Table I, most approaches use an FFT-based algorithm to calculate the respiratory rate. While the FFT-based algorithm demonstrated impressive performance, zero-crossing rate (ZCR) based algorithms require fewer computing resources. In this work, we implement the FFT, ZCR, and Peak-based algorithm by following the InstantRR [6]. We did not consider the Random Forest (RF) and deep learning approach proposed in WearBreathing [8], to restrict computational resource usages on smartphone.

III. System Overview

Figure 1 shows the architecture of our end-to-end *RRMonitor* system. As can be seen in Fig.1, we have three key components: 1) *Data Preparation* for the data recording, transmission, and aggregation of the motion sensor data for breathing signal processing; 2) *Breathing Signal Processing Pipeline* for filtering and processing the earbud motion sensor data to estimate the respiratory rate, and 3) *User-Interface* for displaying the respiratory rate to the user. *RRMonitor* continuously collects data from earbuds and shows the respiration rate to the user in near-real-time. If the user is in a resting position, *RRMonitor* updates the interface every second. We perform several experiments to find the suitable parameters and algorithms to accommodate our design goals. In this section, we describe the key components of *RRMonitor* and define various parameters that need to be adjusted. Our experimentation results are provided in Section IV.

A. Data Preparation

The earbuds are generally kept in the charging box. When the earbuds come out of the charging box, the system starts collecting the sensor data and transmits it to the paired phone using Bluetooth. The system stores the recorded data on the phone until the buffer length is greater than the predefined window length. We provide more details in the following:

1) Recording: We use a Samsung Buds Pro for our *RRMonitor* system, which contains a 3-axis accelerometer and 3-axis gyroscope. Buds Pro also provides a wear status and capable of recording the measurements at an average sampling rate of 80 Hz. However, we experimented with various sampling rates f_s to reduce memory usage.

2) Transmission: As earbuds have minimal memory, the recording needs to be transferred to the phone instantaneously. Therefore, once the recording is started, the system sends the 12-axis measurements (6-axis motion sensors from each earbud) and wear status to the paired phone in small chunks. We can consider that the system sends the recorded data in small chunks. As the system needs to rely on Bluetooth for data transmission, it is expected that some signals will be lost. Earbuds provide a kernel timestamp to handle the data missing issues.

Buds Pro has two separate earbuds: one for the left ear and another for the right ear. However, during our initial exploration, we noticed that the data missing rates are higher when the buds transmit the measurements from both earbuds. Therefore, *RRMonitor* randomly picks the sensor data from one earbud and sends it to the phone.

3) Data Buffers: The breathing module operates on a sliding window of the motion data buffer. We experimented with various window W_s and step sizes S_s for the IMU data buffer and evaluated their impact on accuracy. Since a normal breathing cycle can be between 3–5 seconds, we need at least 2 to 3 cycles to estimate the respiration rate more accurately. Hence, *RRMonitor* has an initialization period in the beginning. The initialization period depends on the window length, as the system needs to populate the whole data window. After that, the system computes the respiration rate at a regular interval, depending on the step size.

B. Breathing Signal Processing Pipeline

For breathing processing pipeline, *RRMonitor* follows a combination of approaches described in SleepMonitor [12], InstantRR [6], and Zephyr [5]. The steps of our pipeline are depicted in the blue shaded region of Figure 1:

1) Window Selection: As explained earlier, some sensor data might get lost during data transmission. Therefore, we check the timestamp difference of each consecutive sample and discard the instances where the timestamp difference is higher than 250ms. If more than 20% data is missing, we drop that particular window. Our system also discard windows if the head movement is high. Similar to SleepMonitor [12], we calculate the total acceleration ($\sqrt{a_x^2 + a_y^2 + a_z^2}$) to estimate the head motion. Since the accelerometer also contains the gravity, the total acceleration should be closer to gravity, when the user is in resting position and motionless. We discard the windows if more than $M\%$ samples are above $10\ m/s^2$. We selected this threshold of $10\ m/s^2$ by following SleepMonitor [12]. We notice that if we keep M smaller, then a large number of windows are discarded. We explore the suitable range for M in Section IV.

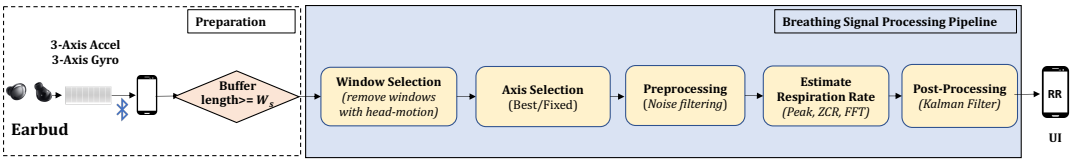

Fig. 1: The *RRMonitor* system architecture that utilizes motion sensors from earbuds to monitor the respiratory rate.

2) Axis Selection: One of the primary goals for the axis selection method is to reduce the memory consumption for the signal preprocessing and respiratory rate extraction process. While both Zephyr [5] and SleepMonitor [12] run their respiratory rate algorithm on a 6-axis motion sensors, it will require significant memories for the whole process. For example, 6-axis motion sensors will consume six times more memory (≈72kb, when the sampling rate is 50Hz) than one-axis. Therefore, in *RRMonitor*, we emphasized more on selecting one axis than 6-axis. We experimented with both best (select axis with the largest periodic variation [6], [7]) and fixed axis (any of the 6-axis) for axis selection.

3) Signal Preprocessing: Once we select the axis for respiratory rate calculation, we first apply a median filter to smooth the signals. Then, we perform z-score normalization on the axis and apply a 5^{th} order Butterworth bandpass with 0.3Hz center frequency [14]. We then apply a 3rd order Savitzky-Golay [15] filter for further smoothing the signal. The filtered signal S_f reduce the noise from the signal and remove the gravity for extracting the respiratory rate.

4) Respiratory Rate Estimation Algorithm: We extract the respiratory rate by following three approaches [6]:

i) **FFT-Based Approach (RR_{fft}):** We first apply the FFT on the filtered signal to compute the maximum frequency component in the selected axis. We then calculate the respiration rate by selecting the frequency of highest amplitude (peak) within 0.13 Hz and 0.66 Hz frequency range (corresponds to 7.8 and 40 BPM).

ii) **Zero-Crossing Based Breathing Rate (RR_{zcr}):** During a breathing cycle, the breathing waveform changes phases while transitioning from inhalation to exhalation phase. Hence, we can easily calculate the breathing rate by identifying the zero-crossing indices. The time between two zero-crossing points generally corresponds to the time it takes to complete a full-breathing cycle (inhalation and exhalation). We take the median time of the breathing cycle to estimate the respiratory rate.

iii) **Peak Detection Based Breathing Rate (RR_{peak}):** We first find the peaks and troughs from the filtered signal. Each trough-to-peak is considered as the inhalation, and peak-to-trough can be considered as the exhalation. Ideally, peaks and troughs must occur in alternating order. To maintain the alternating order, we remove the false peaks and troughs if there are multiple peaks in between two troughs or multiple troughs in between two peaks. We then estimate the respiratory rate by taking the median peak-to-peak distances.

5) Post-Processing: We estimate the quality of our respiratory rate estimation algorithm by taking the standard deviation of three estimations:

$$q_w = \sigma(RR_{fft}, RR_{zcr}, RR_{peak})$$

where q_w denotes the quality of estimation for window W_s. If the signal quality is good enough, the three algorithms should estimate a similar respiratory rate. Therefore, the standard deviation will be lower. On the other hand, if the standard deviation is higher, that indicates that the signal is likely to have motion artifacts. If q_w is below the threshold λ, we consider the estimation as valid.

As we are estimating the respiration rate instantaneously, the system can have some large sudden change in the estimation due to the motion artifacts. While the respiratory rate can have small variations, the sudden change should not be significantly larger than the previous estimation. Therefore, the large sudden changes are likely to be an outlier. We employ a Kalman-Filter based method to remove the outliers by modifying an approach proposed in Zephyr [5]:

$$RR_t = RR_{t-1} + K_t(RR_t - RR_{t-1})$$

where $K_t = \frac{P_{t-1}}{P_{t-1}+q_w}$ is the Kalman Gain and $P_t = (1 - K_t)P_{t-1}$ is the estimation error at time t ($P_0 = 1$). RR_t is the respiration rate at time t.

The novelty of our approach here is that we develop a different quality metric q_w for the Kalman gain calculation, unlike the existing approach [5]. Our approach only depends on single best channel which is more suitable for the resource-constraint environment such as earbuds.

C. User-Interface

We developed an application for showing that displays the respiration rate RR_t in near real-time on a smartphone.

IV. Experimentation

In this section, we present the findings of our experiments to identify the optimal values for the key parameters described in Section III. We performed the evaluations on a dataset collected in a lab study with 30 participants. In the next part of this section, we first discuss our study protocol and then we present the results of our evaluation.

A. Study Protocol

We conducted a lab study with 30 participants (15 male and 15 female, age range: 22-58) in a lab environment, where we asked our participants to wear an earbud during the data collection. Our participants performed several regular and controlled breathing tasks in different postures (sitting, standing, and lying down) during the study. The participants performed the breathing task in a seating position for 1.5 minutes. The duration of the rest of the tasks was for

one minute. Our study was conducted during COVID-19; therefore, the participants wear a mask during the lab study to ensure the safety of other participants and researchers. The study was approved by Institutional Review Board (IRB).

The lab study took approximately 45 mins and facilitated by two researchers using an android application. The researchers used the android app on a Samsung phone to record the timings of the tasks. We used a Samsung Galaxy Buds Pro pair to record the measurements at a 50Hz sampling rate from inertial measurement unit-based sensor units. A Samsung Galaxy S9 was connected with the earbuds and stored all the recordings in the local storage. The recordings were later uploaded to the server.

We collected the ground truth respiratory rate from a Zephyr bioharness chestband that is equipped with a breathing sensor. We recorded the breathing signal at a 20Hz sampling rate. We use a semi-automated algorithm to derive the breathing rate from the breathing waveform. As breathing waveforms are sinusoidal, we use a peak detection algorithm to identify the peaks and troughs on the breathing waveform. One researcher later manually reviewed the identified peaks and troughs on a time-series-based annotation platform. If the automated algorithm missed any peaks/troughs, the researcher manually annotated the missing peaks and troughs. The reference breathing rate is calculated by taking the timestamp difference two consecutive peaks: $60/(ts_p^{i+1} - ts_p^i)$, where ts_p^i denotes the timestamp of i^{th} peak. We needed to discard some reference data for each task due to the low-quality breathing waveforms captured in the chestband.

B. Parameters of Breathing Module

While the algorithms performed better in Python, we report the results from the version implemented in Java. Since the filtering algorithms are slightly different in Java, We choose to use Java to make the results consistent with the on-device performance. We used the same code on the Android application; therefore, the result presented in this section is likely to be consistent with the application.

1) Sampling Rate (fs): We could select one sample rate for our lab study; hence, we conducted an initial experiment by recording the data at 80Hz. From our initial exploration, we found that recording at 50Hz does not significantly impact the mean absolute error (MAE), while it can reduce the memory overhead by 37.5%. Therefore, we select 50 Hz as our sampling rate for recording.

2) Best Algorithm for Respiratory Rate Estimation (RR_t): Table II summarizes the results of different respiratory rate algorithm. As can be seen in Table II, the ZCR algorithm performs the best across all postures (MAE=3.12, W_s=30s) than the FFT-Based Algorithm (MAE= 3.24, W_s=30s). In addition to the performance, the zero-crossing algorithm is extremely lightweight (processing time 0.02ms) than the FFT (1.96ms) and Peak-based (10.28ms) Algorithm (Figure 2). Therefore, we select RR_{zcr} as our main respiratory rate (RR_t) to show it to the user.

3) Window Size Selection (W_s): Previous works [5], [12] have shown that a 30-sec window works best for the respiratory rate algorithm. Since our focus is to save resources for continuous monitoring, we experimented with four different window sizes. As expected, when we decrease the window size, it slightly increases the MAE. For example, if we take a 24-sec window (MAE=3.17) instead of a 30-sec window, the error is raised by $\approx 3.5\%$. Although the processing time does not improve significantly (8.8%), it can save around 14 kb memory for storing the 6-axis accelerometer. Therefore, we select a 24 second data buffer for *RRMonitor*.

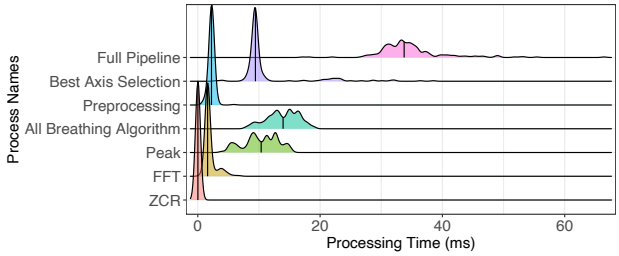

Fig. 2: Processing Time Comparison

4) Step Size Selection (S_s): While small step sizes can provide instantaneous results, it can take more CPU usages. Table II shows the results of one second step size. As can be seen in Table II, *RRMonitor* takes 2-3% CPU load for each one sec interval. While a larger step size could provide a more robust result, a smaller window length is essential for generating the breathing waveform in the app.

5) Channel Selection: As can be seen in Figure 2, the best axis selection algorithm takes a significant amount of processing times (11.31ms). We need to run an FFT algorithm for the 6-axis motion sensors during the best axis selection procedure. Therefore, it takes six times more memory than a fixed axis algorithm. To reduce the computational complexity, we separately ran our algorithm on each of the six axes and evaluated their performance.

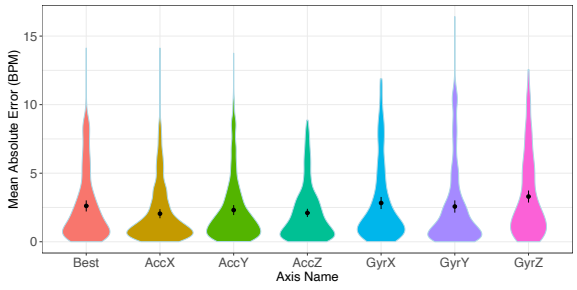

Fig. 3: Error Comparison for Best or Fixed Channel Selection

The performance comparison is depicted in Figure 3. Surprisingly, we found that Acc_x (MAE=2.05) and Acc_z (MAE= 2.10) performs better than the most periodic channel(MAE= 2.61). Unlike previous works [7], the fixed channel-based approach makes more sense to earbuds because earbuds are fixed to the ear, and the Acc_x is the horizontal axis. Therefore, breathing motion affects have higher variation on Acc_x axis. We also analyzed the channels picked by the best channel-based algorithm and found that Acc_x channel is selected as the axis with most periodic variations in

W_s	MAE-Sitting (P=22)				MAE-Standing(P=29)				MAE-Lying (P=26)				Processing Time (Avg)
	N	RR_{fft}	RR_{zcr}	RR_{peak}	N	RR_{fft}	RR_{zcr}	RR_{peak}	N	RR_{fft}	RR_{zcr}	RR_{peak}	
30 secs	530	2.58	**2.33**	3.55	356	3.76	**3.65**	4.80	388	**3.39**	3.53	4.24	37ms
24 secs	604	2.75	**2.61**	3.17	463	3.70	**3.57**	4.37	468	**3.51**	3.52	3.92	34ms
18 secs	710	2.92	**2.90**	3.11	552	3.94	**3.72**	4.02	614	**3.56**	3.70	3.62	27ms
12 secs	674	3.40	**3.26**	3.31	537	3.90	3.78	**3.66**	649	3.94	**3.92**	3.89	22ms

TABLE II: Performance of different respiration rate algorithm under different postures and window size. The result was generated by running the best axis selection procedure. For this set of results, M=50% and $\lambda = 3$. (MAE: Mean Absolute Error, N: Number of valid windows, P: Number of participants with valid ground truth data).

most cases. Therefore, selecting the Acc_x can reduce some computational and memory overhead while improving the accuracy. Therefore, we selected only Acc_x for *RRMonitor*.

6) Motion Artifact Removal: The head motion filtering is also a very crucial process of *RRMonitor*. As earbuds are fixed to the head, a slight head motion can often generate an erroneous result. However, we discard the windows if more than $M\%$ samples are higher than 10 m/s^2. Table III summarizes the performance of head motion filtering. As can be seen in Table III, M=25% gives the best performance while removing one-third of the windows.

M%	MAE (RR_{ZCR})	(%) Windows Removed
5%	1.74	34.3%
10%	1.74	34.3%
25%	1.64	23.2%
50%	2.04	14.1%

TABLE III: Impact on performance with motion filtering

7) Kalman-Filter Based Postprocessing: Our Kalman-Filter-based post-processing algorithm is one of the key contributions of this paper. As we estimate three breathing rates, the prediction itself can be used as a quality control mechanism. Without the Kalman filter, the accuracy can drop by 18% (from 1.64 to 1.92 BPM). We also have parameter λ that is used to filter out erroneous predictions. When λ is high, it will filter fewer erroneous results and can impact the accuracy. For example, changing the λ from 3 to 5 drops the accuracy by 23.2% (MAE : $1.64 \rightarrow 2.02$).

V. Discussions and Conclusions

This paper presents an end-to-end system for respiratory rate monitoring that can run continuously on a mobile device. We envisioned that the user would interact with the application and visualize their breathing waveform in near real-time. Therefore, the critical contribution of this paper is that our system is resource-aware, and we adjusted several key parameters to keep the system lightweight so that the system does not put a significant burden on the existing system. Our analysis shows that our system only takes 2-3% overhead. The overhead can be further reduced by monitoring the breathing rate less frequently and when the user is not interacting with the application.

Respiration rate is a critical vital sign with several health benefits. Our system is the intermediary step for deploying a real-time respiratory algorithm in the wild. As a next step, we plan to evaluate the system with test users and compare the performance. While the current system is limited to estimating the respiration rate at resting conditions, we also plan to expand our approach to respiratory rate estimation while the users perform various activities.

References

[1] S. Rolfe, "The importance of respiratory rate monitoring.," *British journal of nursing (Mark Allen Publishing)*, vol. 28, pp. 504–508, Apr. 2019. Place: England.

[2] M. A. Cretikos, R. Bellomo, K. Hillman, J. Chen, S. Finfer, and A. Flabouris, "Respiratory rate: the neglected vital sign," *Medical Journal of Australia*, vol. 188, no. 11, pp. 657–659, 2008.

[3] C. Kelly, "Respiratory rate 1: why measurement and recording are crucial.," *Nursing Mirror*, vol. 114, pp. 23–24, Apr. 2018.

[4] J. F. Fieselmann, M. S. Hendryx, C. M. Helms, and D. S. Wakefield, "Respiratory rate predicts cardiopulmonary arrest for internal medicine inpatients," *Journal of General Internal Medicine*, vol. 8, no. 7, pp. 354–360, 1993.

[5] H. Aly and M. Youssef, "Zephyr: Ubiquitous accurate multi-sensor fusion-based respiratory rate estimation using smartphones," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, 2016.

[6] M. M. Rahman, E. Nemati, V. Nathan, and J. Kuang, "Instantrr: Instantaneous respiratory rate estimation on context-aware mobile devices," in *13th EAI International Conference on Body Area Networks* (C. Sugimoto, H. Farhadi, and M. Hämäläinen, eds.), (Cham), pp. 267–281, Springer International Publishing, 2020.

[7] J. Hernandez, D. McDuff, and R. W. Picard, "Biowatch: estimation of heart and breathing rates from wrist motions," in *2015 9th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, pp. 169–176, IEEE, 2015.

[8] D. Liaqat, M. Abdalla, P. Abed-Esfahani, M. Gabel, T. Son, R. Wu, A. Gershon, F. Rudzicz, and E. D. Lara, "Wearbreathing: Real world respiratory rate monitoring using smartwatches," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 3, June 2019.

[9] T. Röddiger, D. Wolffram, D. Laubenstein, M. Budde, and M. Beigl, "Towards respiration rate monitoring using an in-ear headphone inertial measurement unit," in *Proceedings of the 1st International Workshop on Earable Computing*, EarComp'19, (New York, NY, USA), p. 48–53, Association for Computing Machinery, 2019.

[10] S. E. Widen, S. Båsjö, C. Möller, and K. Kähäri, "Headphone listening habits and hearing thresholds in swedish adolescents.," *Noise & health*, vol. 19, no. 88, pp. 125–132, 2017.

[11] Y. Nam, B. A. Reyes, and K. H. Chon, "Estimation of respiratory rates using the built-in microphone of a smartphone or headset," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 6, pp. 1493–1501, 2016.

[12] X. Sun, L. Qiu, Y. Wu, Y. Tang, and G. Cao, "Sleepmonitor: Monitoring respiratory rate and body position during sleep using smartwatch," *Proc. ACM Interact. Mob. Wearable Ubiquitous Tech.*, vol. 1, Sept. 2017.

[13] J. Hernandez, Y. Li, J. M. Rehg, and R. W. Picard, "Bioglass: Physiological parameter estimation using a head-mounted wearable device," in *2014 4th International Conference on Wireless Mobile Communication and Healthcare - Transforming Healthcare Through Innovations in Mobile and Wireless Technologies (MOBIHEALTH)*, pp. 55–58, 2014.

[14] B. Porr, *An IIR filter library written in JAVA*, 2021 (accessed Jan, 2021). https://github.com/berndporr/iirj.

[15] S. Wallez, *Savitzky Golay Filter*, 2021 (accessed Jan, 2021). https://github.com/swallez/savitzky-golay-filter.