Mobile/android application for QRS detection using zero cross method

M I Rizqyawan¹, A I Simbolon¹, M A Suhendra¹, M F Amri¹, D E Kusumandari¹
¹UPT BPI, Indonesian Institute of Sciences, Bandung 40135, Indonesia
muhal76@lipi.go.id, arth001@lipi.go.id, agung.fq@gmail.com, mfa001@lipi.go.id, esti_dek@yahoo.com

Abstract. In automatic ECG signal processing, one of the main topics of research is QRS complex detection. Detecting correct QRS complex or R peak is important since it is used to measure several other ECG metrics. One of the robust methods for QRS detection is Zero Cross method. This method uses an addition of high-frequency signal and zero crossing count to detect QRS complex which has a low-frequency oscillation. This paper presents an application of QRS detection using Zero Cross algorithm in the Android-based system. The performance of the algorithm in the mobile environment is measured. The result shows that this method is suitable for real-time QRS detection in a mobile application.

1. Introduction
Electrocardiogram (ECG) signal is a signal produced by measuring electrical activities from heart's muscle. It is a non-invasive instrument and due to this characteristic of ECG, it is widely used for cardiovascular diseases. It is also one of most popular research field in biomedical signal processing, with applications ranging from automatic detection of arrhythmia [1-3], sleep apnea [4, 5], to psychological stress [6, 7].

In automatic ECG signal processing, one of the main topics of research is QRS complex detection. This area of research has been explored for more than three decades [8]. Detecting correct QRS complex or R peak is important since it is used to measure several other ECG metrics, such as RR Interval, QRS Width, and Heart Rate Variability (HRV). These metrics are often used as features for ECG classification. The most common method for QRS detection is Pan Tomkins method [8] which use an adaptive threshold after the signal is processed with a bandpass filter, derivative, squaring, and moving average. Zero Cross method [9] is another QRS detection method which employed an adaptive filtering in the steps of its processing. This method uses an addition of high-frequency signal and zero crossing count to detect QRS complex which has a low-frequency oscillation. By employing the zero-crossing count, this method is robust against noise and performs well in the noisy ECG signal.

On the other hand, the other topic of research in ECG is the application of automatic ECG signal processing. Most of the study is intended to be applied in workstation/desktop computer. Some of the research is exploring the application of the techniques in hardware [10] (such as embedded system), and also mobile phones [11-14]. The exploration of ECG analysis in mobile phones is increasing due to the increase in availability of smartphones. Smartphones become a ubiquitous computer, with increasing capabilities in the terms of computation power. Although the computation power of today’s smartphone is clearly higher than computer several decades ago, however, there's always a limitation to the mobile phone, which cannot have a full processing power and memory of a desktop computer in
the same generation. Therefore, further study needed in the implementation of mobile ECG analysis to make the processing fast enough and feasible in real-time use-case.

2. Methodology
This research aims at developing a smart classifier of arrhythmia for android device. The research consists of two sub-research focuses, the classifier model development and development and implementation of the smart system in android device. The block diagram of our research can be seen in figure 1. As can be seen in figure 1, this paper falls on the development of the android smart system app, with the focus on QRS detection module. In this paper, our aim is to explore whether the algorithm fits to be applied in the mobile environment or not. The other sub-research focus such as classification and system architecture is discussed in our other paper [15, 16].

In this experiment, we use MIT-BIH arrhythmia database [17] as our testing data. Single channel (MLII channel) is used for the experiment. In this experiment, we use selected records of the database denotes as dataset 1 (ds1) in protocol proposed by Chazal et. al. [1]. The raw data is exported to CSV file, which will be loaded into the Android app later. In order to examine the computation load required by the algorithm, processing time of QRS detection is measured. The processing time is measured by setting up a timer just before and after the process of QRS detection using Java's nanoTime. The processing time is calculated by subtracting the end timer with start timer. The flowchart of the process is shown in figure 2.

3. Zero Cross Algorithm
Zero Crossing Count for QRS detection is proposed by Kohler et. al at 2003 [9] with the aim to develop a QRS detection algorithm with low computational load and high detection performance. The algorithm can be divided into three main blocks; feature extraction, event detection, and temporal localization. The feature extraction aims to generate feature signal, which would be used as the input of event detection process. The event detection block searches the possible location of QRS complex by comparing the feature signal with the threshold. Temporal localization searches the actual peak location in the block of event detected in the previous process.

3.1 Feature Extraction
This process aims to prepare the signal and process it all the way to the counting of the zero crossing of the signal to the x-axis. The first process of the feature extraction is filtering. Bandpass filter with cut-off frequency of 18 Hz and 35 Hz are used. Then the next process is the nonlinear transform, by squaring the signal multiplied by its signum. The process can be written as:

\[ y(n) = \text{sign}(x_f(n)) \cdot x^2_f(n) \]  

(1)

where \( y(n) \) is the transformed signal and \( x_f(n) \) denotes the bandpass filtered signal. Using this process, the important features of the signal can be distinguished from the rest of other features.
This paper focuses on Android application development, especially on QRS Detection algorithm application.

The next process is to add the high-frequency sequence to the nonlinear transformed signal. To do so, the amplitude of the high-frequency sequence must be calculated first. The estimated amplitude of the high-frequency sequence is determined by:

$$K(n) = \lambda_K K(n-1) + (1 - \lambda_K) |y(n)c|$$  \hspace{1cm} (2)

where $K(n)$ is the estimated amplitude, $\lambda_K$ is the forgetting factor, $y(n)$ is the nonlinear transformed signal, and $c$ denotes the constant gain. After the amplitude is estimated, then we can calculate the high-frequency sequence by:

$$b(n) = (-1)^n \cdot K(n)$$  \hspace{1cm} (3)
where \( b(n) \) is the high-frequency sequence and \( K(n) \) is the estimated amplitude. The high-frequency sequence is added to the nonlinear transformed signal using:

\[
z(n) = y(n) + b(n)
\]

where \( z(n) \) is the signal after high-frequency addition, \( y(n) \) is the nonlinear transformed signal, and \( b(n) \) is the high-frequency sequence. At this point, the signal is ready to be processed at the next step, which is zero crossing count. The zero crossing is counted by using autoregressive low pass filter:

\[
D(n) = \lambda_D D(n-1) + (1 - \lambda_D) d(n)
\]

where \( d(n) \) is the count of zero-crosses at the segment and \( \lambda_D \) is forgetting factor.

### 3.2 Event Detection

This process provides the suspected area of QRS complex by comparing the feature signal with an adaptive threshold. The adaptive threshold is computed by:

\[
\Theta(n) = \lambda_{\Theta} \Theta(n-1) + (1 - \lambda_{\Theta}) D(n)
\]

where \( \lambda_{\Theta} \) forgetting factor, \( D(n) \) is the feature signal and \( \Theta(n) \) is the adaptive threshold. When feature signal is lower than the adaptive threshold, the event is marked as the start. The algorithm continued the scanning process, and when the feature signal is higher than the threshold, it is marked as the end. However, if the occurrence of start event is lower than time-out value, the two events are merged into one as it likely comes from one QRS complex.

### 3.3 Temporal Localization

This process searches the location of the R-wave in the interval provided by the previous process. To locate the R-wave, the algorithm search for the maximum and minimum value of the signal in the suspected interval of QRS complex. If the value of the maximum is lower than the absolute value of the minimum, then the R-wave is the temporal location of minimum value. But if not, the R-wave is determined by the temporal location of maximum value. To point the actual location of the peak in the raw signal, please note that the delay caused by the filtering needs to be considered.

### 4. Result

The zero cross algorithm is implemented using Java as the programming language, which is the official development language for android application development. The screenshot of the app can be seen in figure 3. To test the processing time of the algorithm, three phones are used which represents each category/class of smartphone. Motorola Moto E4, Oppo F1s, and Xiaomi Mi5 which represents low-end, middle-end, and high-end class Android phone respectively. Table 1 shows the result of processing time test using three different smartphones mentioned above.

**Table 1. Processing time test on different devices.**

| Device                | Min (ms) | Max (ms) | Average (ms) |
|-----------------------|----------|----------|--------------|
| Motorola Moto E4      | 14.85    | 33.5     | 15.7         |
| Oppo F1s              | 270.0    | 426.1    | 299.8        |
| Xiaomi Mi5            | 120.4    | 131.3    | 124.3        |

To measure the real-time capability of the method, the result from table 1 is converted to real-time factor, as proposed by [11] using:

\[
R_f = \frac{P_t}{MeasurementWindow}
\]
where $R_t$ is real-time factor value, and $P_t$ is processing time acquired from table 1. Figure 4 shows the minimum, maximum, and average real-time factor required for the algorithm to run on the tested devices.

![SmartECG screenshot](image1)

**Figure 3.** The screenshot of the application.

![Average Real-time Factor on Various Devices](image2)

**Figure 4.** The minimum, maximum, and average real-time factor required for the algorithm to run on the tested devices.

5. **Conclusion**

This paper presents an application of QRS detection using Zero Cross algorithm in the Android-based system. The performance of the algorithm in the mobile environment is measured using three different devices as the test-bed.

The result shows that this method is suitable for real-time QRS detection in a mobile application. The average real-time factor of processing time in Xiaomi Mi5 is 0.0022, which is much less than 1. This result is much expected as the Xiaomi Mi5 presents the high-end class smartphone. The Oppo F1s shows the longest time required for processing. The maximum real-time factor of Oppo is highest than the other devices at 0.0071. But at this highest value, the result shows that the real-time factor is much less than 1. The surprising result comes from Motorola Moto E4 which obtained the lowest processing time and real-time factor required by the algorithm to run. This is surprising as the Moto E4 represents lower-end smartphone. Further study is required to investigate this result, but generally, the result shows that this method is feasible to be used in real-time QRS detection.

**Acknowledgments**

This research was supported by the Insinas Ristek funded by Minister of Research, Technology, and Higher Education, Indonesia.

**References**

[1] deChazal P, O'Dwyer M and Reilly R 2004 Automatic classification of heartbeats using ECG morphology and heartbeat interval features *IEEE Transactions on Biomedical Engineering*, 51(7) pp 1196-1206

[2] Luz E, Schwartz W, Cámara-Chávez G and Menotti D 2016 ECG-based heartbeat classification for arrhythmia detection: a survey *Computer Methods and Programs in Biomedicine* 127 pp 144-164
[3] Escalona-Moran M, Soriano M, Fischer I and Mirasso C 2015 Electrocardiogram classification using reservoir computing with logistic regression IEEE Journal of Biomedical and Health Informatics 19(3) pp 892-898

[4] Hassan A and Haque M 2016 Computer-aided obstructive sleep apnea screening from single-lead electrocardiogram using statistical and spectral features and bootstrap aggregating Biocybernetics and Biomedical Engineering 36(1) pp 256-266

[5] Hassan A and Haque M 2017 An expert system for automated identification of obstructive sleep apnea from single-lead ECG using random under sampling boosting Neurocomputing 235 pp 122-130

[6] Lampert R 2015 ECG signatures of psychological stress Journal of Electrocardiology 48(6) pp 1000-05.

[7] Han L, Zhang Q, Chen X, Zhan Q, Yang T and Zhao Z 2017 Detecting work-related stress with a wearable device Computers in Industry 90 pp 42-49

[8] Pan J and Tompkins W 1985 A real-time QRS detection algorithm IEEE Transactions on Biomedical Engineering 32(3) pp 230-236

[9] Köhler B, Hennig C and Orglmeister R 2003 QRS detection using zero crossing counts Progress in Biomedical Research 8(3)

[10] Zhang C and Bae T 2012 VLSI friendly ECG QRS complex detector for body sensor networks IEEE Journal on Emerging and Selected Topics in Circuits and Systems 2(1) pp 52-59

[11] Sufi F, Fang Q and Cosic I 2007 ECG R-R peak detection on mobile phones 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society

[12] Elgendi M, Eskofier B, Dokos S and Abbott D 2014 Revisiting QRS detection methodologies for portable, wearable, battery-operated, and wireless ECG systems PLoS ONE 9(1) p. e84018

[13] Gradl S, Kugler P, Lohmuller C and Eskofier B 2012 Real-time ECG monitoring and arrhythmia detection using Android-based mobile devices Annual International Conference of the IEEE Engineering in Medicine and Biology Society

[14] Leutheuser H, Gradl S, Kugler P, Anneken L, Arnold M, Achenbach S and Eskofier B 2014 Comparison of real-time classification systems for arrhythmia detection on Android-based mobile devices 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society

[15] Rizqyawan M, Simbolon A and Kusumandari D Weighted SVM with RR interval based features for Android-based arrhythmia classifier

[16] Rizqyawan M, Amri M, Pratama R and Turnip A 2016 Design and development of Android-based cloud ECG monitoring system 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)

[17] Moody G and Mark R 2001 The impact of the MIT-BIH arrhythmia database IEEE Engineering in Medicine and Biology Magazine 20(3) pp 45-50