УДК 004.021
DOI https://doi.org/10.32782/IT/2021-2-1

Андрій ВОРОПАЙ
аспірант, Дніпровський національний університет імені Олеся Гончара, просп. Гагаріна, 72, м. Дніпро, Україна, 49010, voropayandrey@gmail.com
***ORCID:** 0000-0002-2783-6700*

Володимир САРАНА
кандидат технічних наук, доцент кафедри телекомунікаційних мереж та систем, Дніпровський національний університет імені Олеся Гончара, просп. Гагаріна, 72, м. Дніпро, Україна, 49010, vsarana@ukr.net
***ORCID:** 0000-0002-7778-3176*

Бібліографічний опис статті: Воропай, А., Сарана, В. (2021). Алгоритм пошуку R-зубців у реальному часі для ЕКГ сигналу з підвищеним рівнем шуму). *Information Technology: Computer Science, Software Engineering and Cyber Security*, 2, 3–9, doi: https://doi.org/10.32782/IT/2021-2-1

АЛГОРИТМ ПОШУКУ R-ЗУБЦІВ У РЕАЛЬНОМУ ЧАСІ ДЛЯ ЕКГ СИГНАЛУ З ПІДВИЩЕНИМ РІВНЕМ ШУМУ

У роботі було розглянуто існуючі методи та алгоритми пошуку R-зубців у ЕКГ сигналі, проаналізовано роботу цих методів та алгоритмів на сигналі з підвищеним рівнем шуму. Також було розглянуто різні типи та природу утворення шумів в ЕКГ сигналі та методи їх зменшення. У результаті цієї роботи було розроблено власний алгоритм пошуку QRS комплексу та R-зубців у реальному часі. ***Метою роботи*** *є розробка алгоритму пошуку R-зубців у реальному часі для ЕКГ сигналу, записаного з нестандартних відведень. Реалізація поставленої мети передбачає запис ЕКГ сигналу з поверхні голови людини, фільтрацію сигналу та розробку алгоритму детектування QRS комплексу.* ***Методологія*** *вирішення поставленого завдання полягає в пошуку шаблону QRS комплексу у диференційному сигналі згідно з нахилом фронтів R-зубця на трьох точках та інтервалу між цими точками.* ***Наукова новизна.*** *Запропонований алгоритм дозволяє значно покращити пошук QRS комплексу з ЕКГ сигналу з підвищеним рівнем шуму. Представлений алгоритм не потребує значних обчислювальних потужностей, тому може використовуватись в натільних пристроях з малопотужними мікроконтролерами для підвищення точності замірів серцевої діяльності у повсякденному житті.* ***Висновки****. Отримані результати та алгоритм можуть бути використані для подальшої розробки та досліджень. Всі отримані результати представлені в графічному вигляді з детальним описом в даній роботі.*

Ключові слова: *ЕКГ, QRS-комплекс, алгоритми, фільтрація.*

Andrii VOROPAI
Master's degree, PhD student, Oles Honchar Dnipro National University, 72 Haharina ave., Dnipro, Ukraine, 49010, voropayandrey@gmail.com
***ORCID:** 0000-0002-2783-6700*

Volodymyr SARANA
Candidate of Technical Sciences, Associate Professor of Department of Electronic Means of Telecomminications, Oles Honchar Dnipro National University, 72 Haharina ave., Dnipro, Ukraine, 49010, vsarana@ukr.net
***ORCID:** 0000-0002-7778-3176*

To cite this article: Voropai, A., Sarana, V. (2021). Alhorytm poshuku R-zubtsiv u realnomu chasi dlia EKH syhnalu z pidvyshchenym rivnem shumu [Real-time r-peak detection algorithm for low SNR ECG signal]. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 2, 3–9, doi: https://doi.org/10.32782/IT/2021-2-1

REAL-TIME R-PEAK DETECTION ALGORITHM FOR LOW SNR ECG SIGNAL

The aim *of the article is the research and development of an algorithm for finding R-teeth in real time for the ECG signal recorded from non-standard leads. The realization of this goal involves recording the ECG signal from the surface of the human head, signal filtering and development of an algorithm for detecting the QRS*

*complex. **Scientific novelty**. The proposed algorithm can significantly improve the search for QRS complex from the ECG signal with a high noise level. The presented algorithm does not require significant computing power, so it can be used in body devices with low-power microcontrollers to improve the accuracy of heart rate measurements in everyday life. **Conclusions.** The obtained results and algorithm can be used for further development and research. All the obtained results are presented in graphical form with a detailed description in this paper.*

***Key words:** ECG, QRS-complex, algorithms, filtering.*

Актуальність проблеми. Згідно з Всесвітньою організацією охорони здоров'я, серцево-судинні захворювання (ССЗ) є найпоширенішою причиною смертності у людей. В 2016 році 31% усіх випадків смертей були спричинені ССЗ [1]. Кожен рік зростає кількість смертей від хвороби серця, цей факт вимагає уваги до розробки методів попередження небезпечного стану серцево-судинної системи людей. Таких як: цілодобовий моніторинг електричної активності серця (холтерівське моніторування) та вбудовані засоби реєстрації ЕКГ у повсякденні прилади та речі. Останнє вимагає розробку нових методів та алгоритмів аналізу та пошуку корисної інформації у ЕКГ сигналі з великим вмістом шуму та артефактів.

Аналіз останніх досліджень і публікацій. Основним показником діяльності серцево-судинної системи є кількість ударів серця за хвилину та варіабельність цього показника з часом. Для отримання кількості ударів за хвилину потрібно знайти R-зубець або QRS комплекс, що відповідає за деполяризацію шлуночків серця. Де-факто стандартом алгоритму пошуку R-зубців є алгоритм Pan-Tompkins, розроблений у 1985 році науковцями Jiapu Pan та Willis J. Tompkins [2]. Цей алгоритм у різних модифікаціях широко використовується по цей день попри той факт, що ефективність цього методу значно падає при великому вмісті шуму та артефактів у вхідному сигналі (менше 12 dB SNR) [3]. Шум під час вимірювання є результатом багатьох факторів, таких як: неміцний контакт між електродами та шкірою, м'язовий шум, перешкоди лінії електропередач та інших приладів, а також загальний рух користувача [4]. Існують методи адаптивної фільтрації [5], навіть використання нейронних мереж [6], але ці методи боротьби з шумами потребують великих обчислювальних потужностей, що не підходить для невеликих натільних систем з дуже обмеженою потужністю та часом роботи на одному заряді акумулятора.

Мета статті. На підставі дослідженого та проаналізованого матеріалу наукових досліджень було запропоновано алгоритм пошуку QRS комплексу у ЕКГ сигналі з низькими показниками сигнал-шум, а також реалізацію цього алгоритму на мові Python. Основна мета статті полягає в описі розробленого методу та опису можливостей його подальшого використання у вбудованих натільних системах для пошуку QRS комплексу у сигналі з великим рівнем шуму при обмежених ресурсах обчислювальних систем.

Виклад основного матеріалу. Існуючі алгоритми пошуку QRS комплексу мають дві основні категорії, перша – це прості алгоритми, які дуже часто засновані на алгоритмі Pan-Tompkins [7], вони можуть бути легко реалізовані у вбудованих системах з обмеженими ресурсами, але не можуть працювати з потрібним рівнем точності при появі шуму, що знижує рівень сигнал-шум до менше, ніж 12 dB. Друга основна категорія алгоритмів заснована на частотному аналізі, адаптивних фільтрах та нейронних мережах, що потребують значних ресурсів обчислювальних систем, які технічно та матеріально складно вбудовувати у натільні пристрої. У цій роботі було спробувано об'єднати переваги двох категорій алгоритмів в один.

Класичний алгоритм Pan-Tompkins складається з 6 етапів (Рис. 1):

1) Фільтрування вхідного сигналу вузьким смуговим фільтром;
2) Диференціація сигналу, x(n) – x (n – 1);
3) Піднесення до ступеню, pow(x(n), 2);
4) Сумація отриманого сигналу рухомим вікном;
5) Пошук максимального та мінімального значення сигналу для побудови порогового значення;
6) Маркування сигналу, що перевищує порогове значення, як R-зубець.

Звісно, при такому методі можна дуже легко та швидко знайти QRS комплекс, але тільки якщо сигнал-шум запису перевищує 12 dB, в іншому випадку велика ймовірність детектування артефактів та шуму, як R-зубці. Щоб уникнути цих проблем було розроблено власних алгоритм, який складається 8 етапів (Рис. 2):

1) Фільтрація вхідного сигналу смуговим фільтром Баттерворта з частотами зрізу 5 та 35 Гц (Рис. 3);
2) Диференціація фільтрованого сигналу;

Рис. 1. Структурна схема Pan Tompkins алгоритму пошуку R-зубців

Рис. 2. Структурна схема розробленого алгоритму пошуку R-зубців

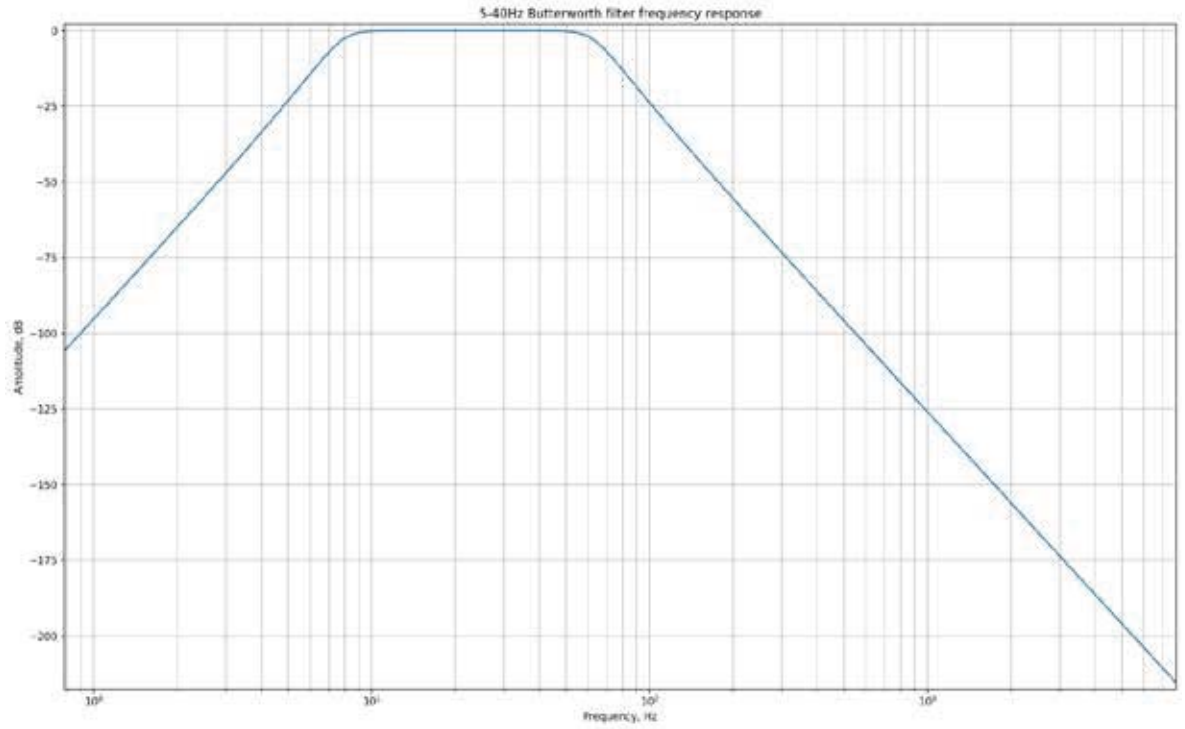

Рис. 3. Амплітудно-частотна характеристика використаного смугового фільтру

3) Піднесення до третього ступеню для збереження знаку;

4) Розрахунок порогових значень для позитивних та негативних чисел;

5) Фіксація порогового значення при знаходженні позитивного числа більшого, ніж порогове;

6) Пошук трьох точок, дві позитивні, одна негативна по центру з такими обмеженими

інтервалами: Tqrs = 40-120 мс, Tqr = Trs = 20-60 мс (Рис. 4);

7) Фільтрування знайдених QRS комплексів по амплітудним ознакам, амплітуда негативної точки повинна перевищувати амплітуду позитивних точок;

8) Остаточна валідація знайденого комплексу.

Метод, описаний вище, здатний працювати з вхідним сигналом з рівнем шуму нижче 12 dB, при цьому без помилкового детектування артефактів (Рис. 5).

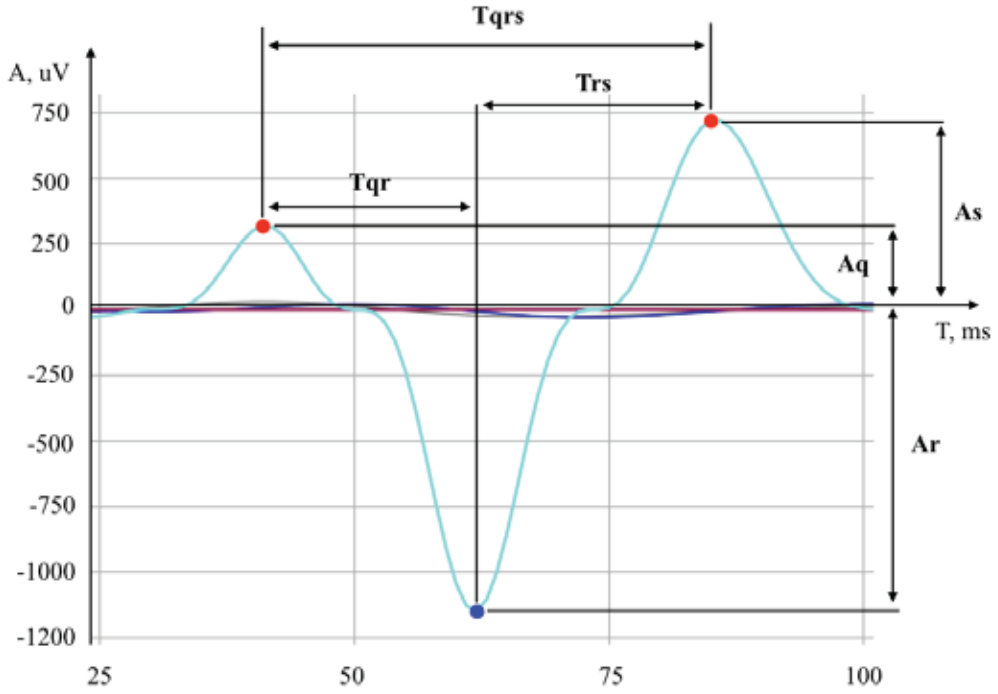

Рис. 4. Диференціальний сигнал та опис основних ознак QRS комплексу

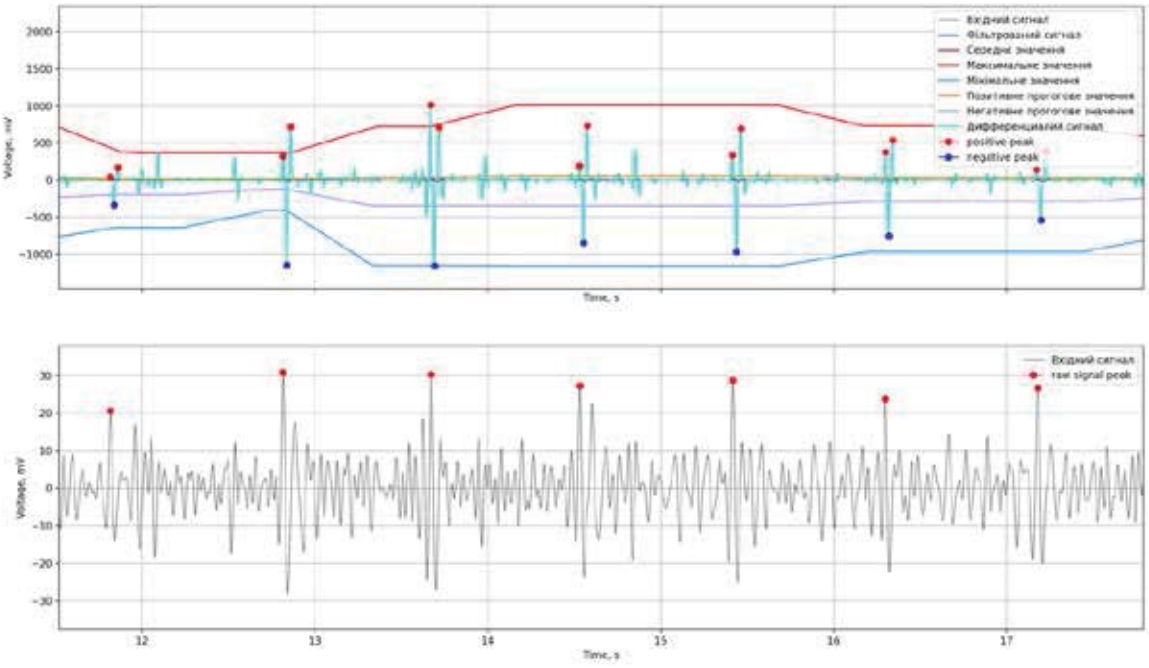

Рис. 5. Вигляд вхідного сигналу з низьким рівнем сигнал-шум

Реалізація алгоритму мовою Python:

```
  def update(self, sample):
          filtered_sample = self.filter5_40.filter_sample(sample)
          diff_sample = -(self.last_filtered_sample - filtered_sample)
          squared_sample = pow(diff_sample, 3)

          self.moving_window = np.roll(self.moving_window, -1)
          self.moving_window[self.moving_window_length - 1] = squared_sample

          final_signal_amplitude = squared_sample

          self.final_signal_list.append(final_signal_amplitude)

          mean = np.mean(self.moving_window)
          max = np.max(self.moving_window)
          min = np.min(self.moving_window)

          self.mean_average_array = np.roll(self.mean_average_array, -1)
          self.mean_average_array[self.average_array_size - 1] = mean

          self.max_average_array = np.roll(self.max_average_array, -1)
          self.max_average_array[self.average_array_size - 1] = max

          self.min_average_array = np.roll(self.min_average_array, -1)
          self.min_average_array[self.average_array_size - 1] = min

          average_mean = np.average(self.mean_average_array)
          average_max = np.average(self.max_average_array)
          average_min = np.average(self.min_average_array)

          positive_amplitude = average_max - average_mean
          negative_amplitude = average_mean + average_min

          positive_amplitude_threshold = average_mean + (positive_amplitude * self.positive_
threshold_coefficient)
          negative_amplitude_threshold = average_mean + (negative_amplitude * self.negative_
threshold_coefficient)

          if final_signal_amplitude >= positive_amplitude_threshold and self.is_positive_
threshold_reached == False:
              # First fix the amplitude threshold
              self.fixed_positive_threshold = positive_amplitude_threshold
              self.is_positive_threshold_reached = True
              self.positive_peak_start_index = self.global_index

          if final_signal_amplitude <= negative_amplitude_threshold and self.is_negative_
threshold_reached == False:
              # First fix the amplitude threshold
              self.fixed_negative_threshold = negative_amplitude_threshold
              self.is_negative_threshold_reached = True
              self.negative_peak_start_index = self.global_index

          if self.is_positive_threshold_reached == True:
              if final_signal_amplitude < self.fixed_positive_threshold:
                  self.is_positive_threshold_reached = False
                  max_index = self.find_max(self.final_signal_list, self.positive_peak_start_
index, self.global_index)
```

```
                    self.positive_slope_peak_list.append(max_index)
                    if len(self.negative_slope_peak_list) > 0 and len(self.positive_slope_peak_
list) > 1:
                        last_negative_peak_index = self.negative_slope_peak_list[len(self.
negative_slope_peak_list) - 1]
                        previous_positive_peak_index = self.positive_slope_peak_list[len(self.
positive_slope_peak_list) - 2]
                        current_positive_peak_index = max_index
                        current_positive_to_last_negative_ms = ((current_positive_peak_index -
last_negative_peak_index) / self.data_frame.get_sampling_frequency()) * 1000
                        if PhysiologyLimits.RS_INTERVAL_DURATION_MAX_MS >= current_positive_to_
last_negative_ms >= PhysiologyLimits.RS_INTERVAL_DURATION_MIN_MS:
                            current_positive_to_previous_positive_ms = ((current_positive_peak_
index - previous_positive_peak_index) / self.data_frame.get_sampling_frequency()) * 1000
                            if PhysiologyLimits.QR_INTERVAL_DURATION_MAX_MS + PhysiologyLimits.
RS_INTERVAL_DURATION_MAX_MS >= current_positive_to_previous_positive_ms >= PhysiologyLimits.
QR_INTERVAL_DURATION_MIN_MS + PhysiologyLimits.RS_INTERVAL_DURATION_MIN_MS:
                                #  We found a QRS complex
                                if abs(self.final_signal_list[last_negative_peak_index]) >
self.final_signal_list[previous_positive_peak_index] and abs(self.final_signal_list[last_
negative_peak_index]) > self.final_signal_list[current_positive_peak_index]:
                                    self.negative_slope_peak_list.append(last_negative_peak_
index)

                                    self.positive_slope_marker_list.append(previous_positive_
peak_index)
   self.positive_slope_marker_list.append(current_positive_peak_index)
                                    self.negative_slope_marker_list.append(last_negative_peak_
index)

   self.final_peak_index_list.append(last_negative_peak_index)
                                    raw_signal_peak_index = self.find_max(self.history_1.raw_
samples_history, (last_negative_peak_index - int((PhysiologyLimits.QR_INTERVAL_DURATION_MAX_MS /
1000) * self.data_frame.get_sampling_frequency())), last_negative_peak_index)
                                    self.raw_signal_peak_list.append(raw_signal_peak_index)


           if self.is_negative_threshold_reached == True:
               if final_signal_amplitude > self.fixed_negative_threshold:
                   self.is_negative_threshold_reached = False
                   max_index = self.find_min(self.final_signal_list, self.negative_peak_start_
index, self.global_index)
                   self.negative_slope_peak_list.append(max_index)
           self.last_filtered_sample = filtered_sample
     self.global_index += 1
```

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямку. В ході виконання даного дослідження було створено оригінальний алгоритм автоматичного пошуку QRS комплексу у ЕКГ сигналі з великим вмістом шумів різного роду. Для перевірки теорії було реалізовано даний алгоритм на мові Python з використанням таких бібліотек, як signal, numpy, matplotlib та інші.

ЛІТЕРАТУРА:

1. World Health Organization, Cardiovascular diseases (CVDs). URL: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

2. Jiapu Pan and Willis J. Tompkins, A Real-Time QRS Detection Algorithm, IEEE Transactions On Biomedical Engineering, Vol. Bme-32, No. 3, March 1985. DOI: 10.1109/TBME.1985.325532

3. M. A. Z. Fariha, R. Ikeura 2, S. Hayakawa 2, S. Tsutsumi. Analysis of Pan-Tompkins Algorithm Performance with Noisy ECG Signals, 2020 J. Phys.: Conf. Ser. 1532 012022. DOI: 10.1088/1742-6596/1532/1/012022

4. Primož Lavricˇ, Matjaž Depolli, Robust beat detection on noisy differential ECG, 2016. DOI: 10.1109/MIPRO.2016.7522172

5. Minseok Seo, Minho Choi, Jun Seong Lee and Sang Woo Kim, Adaptive Noise Reduction Algorithm to Improve R Peak Detection in ECG Measured by Capacitive ECG Sensors, 2018. DOI: 10.3390/s18072086

6. Yande Xiang, Zhitao Lin and Jianyi Meng, Automatic QRS complex detection using two-level convolutional neural network, 2018. DOI: 10.1186/s12938–018–0441–4

7. Hooman Sedghamiz, Matlab Implementation of Pan Tompkins ECG QRS Detector, 2014. DOI: 0.13140/RG.2.2.14202.59841

REFERENCES:

1. World Health Organization, Cardiovascular diseases (CVDs). URL: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

2. Jiapu Pan and Willis J. Tompkins, A Real-Time QRS Detection Algorithm, IEEE Transactions On Biomedical Engineering, Vol. Bme-32, No. 3, March 1985. DOI: 10.1109/TBME.1985.325532

3. M. A. Z. Fariha, R. Ikeura 2, S. Hayakawa 2, S. Tsutsumi. Analysis of Pan-Tompkins Algorithm Performance with Noisy ECG Signals, 2020 J. Phys.: Conf. Ser. 1532 012022. DOI: 10.1088/1742-6596/1532/1/012022

4. Primož Lavricˇ, Matjaž Depolli, Robust beat detection on noisy differential ECG, 2016. DOI: 10.1109/MIPRO.2016.7522172

5. Minseok Seo, Minho Choi, Jun Seong Lee and Sang Woo Kim, Adaptive Noise Reduction Algorithm to Improve R Peak Detection in ECG Measured by Capacitive ECG Sensors, 2018. DOI: 10.3390/s18072086

6. Yande Xiang, Zhitao Lin and Jianyi Meng, Automatic QRS complex detection using two-level convolutional neural network, 2018. DOI: 10.1186/s12938–018–0441–4

7. Hooman Sedghamiz, Matlab Implementation of Pan Tompkins ECG QRS Detector, 2014. DOI: 0.13140/RG.2.2.14202.59841