An Event-Based System for Low-Power ECG QRS Detection

Silvio Zanoli[1], Tomas Teijeiro[1], Fabio Montagna[2], and David Atienza[1]
[1]Embedded Systems Laboratory (ESL), EPFL, Lausanne, Switzerland
[2]DEI, University of Bologna, Bologna, Italy

Abstract—**One of the greatest challenges in the design of modern wearable devices is energy efficiency. While data processing and communication have received a lot of attention from the industry and academia, leading to highly efficient microcontrollers and transmission devices, sensor data acquisition in medical devices is still based on a conservative paradigm that requires regular sampling at the Nyquist rate of the target signal. This requirement is usually excessive for signals that are typically sparse and highly non-stationary, leading to data overload and a waste of resources in the full processing pipeline. In this work we propose a new system to create event-based heart-rate analysis devices, including a novel algorithm for QRS detection that is able to process electrocardiogram signals acquired irregularly and much below the theoretically-required Nyquist rate. This technique allows us to drastically reduce the average sampling frequency of the signal and, hence, the energy needed to process it and extract the relevant information. We implemented both the proposed event-based algorithm and a state-of-the-art version based on regular sampling on an ultra-low power hardware platform, and the experimental results show that the event-based version reduces the energy consumption in runtime up to 15.6 times, while the detection performance is maintained at an average F1 score of 99.5%.**

Index Terms—**event-based sampling, IoT, low-power biosignal processing, gQRS, ECG QRS detection.**

I. Introduction

Continuous bio-signal monitoring through Wireless Body Sensor Nodes (WBSN), in combination with signal processing and machine learning techniques, are guiding a paradigm change in health surveillance in different scenarios, like the activity tracking of the general population or the follow-up of patients with chronic diseases. But as the number of users scales to hundreds of thousands or even to millions, a number of problems arise, mainly related to the management of the large amounts of generated data and the energy efficiency. In the last years, the pursuit of smaller, more efficient wearable devices and with higher battery life has led to different optimizations in the signal acquisition, processing, and communication stages. One of this optimizations is the adoption of a non-uniform sampling scheme, which can reduce the amount of data to be captured, processed, transmitted and stored [1]. The sparse nature of many bio-signals, like the electrocardiogram (ECG), makes it possible to go beyond the classical Nyquist-Shannon sampling theorem and greatly reduce the global sampling rate of the signal without losing significant information. For

This work has been funded by the European Comission through the Human Brain Project (HBP) SGA2 (GA No. 785907) and the DeepHealth Project (GA No. 825111), and by the SNF through the ML-Edge Project (GA No. 182009).

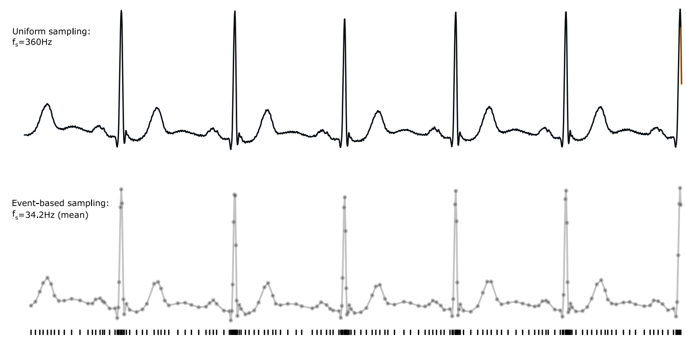

Fig. 1. Uniform vs non-uniform sampled ECG segment. Black lines at the bottom represent the sampling density

this, techniques such as compressed sensing [2] or activity detection [3] can be used. An example is shown in in Fig. 1, where we can see a typical ECG signal sampled uniformly and the event-based samples taken by focusing on the areas with relevant activity. As we can notice, the sampling frequency varies according to the frequency of the signal.

However, this sampling scheme prevents the use of most of the existing biosignal processing algorithms, which assume a uniform sampling frequency. Therefore, the objective of this work is to start filling this gap by developing an algorithm for heart-beat detection, and more specifically for the QRS complex [4], in non-uniformly sampled ECG signals, and implementing it in an actual low-power hardware set-up. This task is of particular interest because it gives us the main tool for heart rate analysis, which is the first and primary task done in ECG processing. Such tool can be used as the fundamental building block to develop a variety of applications like arrhythmia monitoring, QRS delineation, P and T wave detection and classification, and many others [5].

The work can be divided into three main topics: the adopted event-based sampling technique, the adaptation of a state-of-the-art heart-beat detection algorithm to this new environment, and then the implementation on a low-power Micro Controller Unit (MCU), in order to make it ready for a practical implementation in a real embedded wearable device. Finally, the performance and energy efficiency of our solution is evaluated and compared with the state-of-the art.

II. Materials and methods

Figure 2 shows a high-level architecture of the proposed implementation, which is described in the following section.

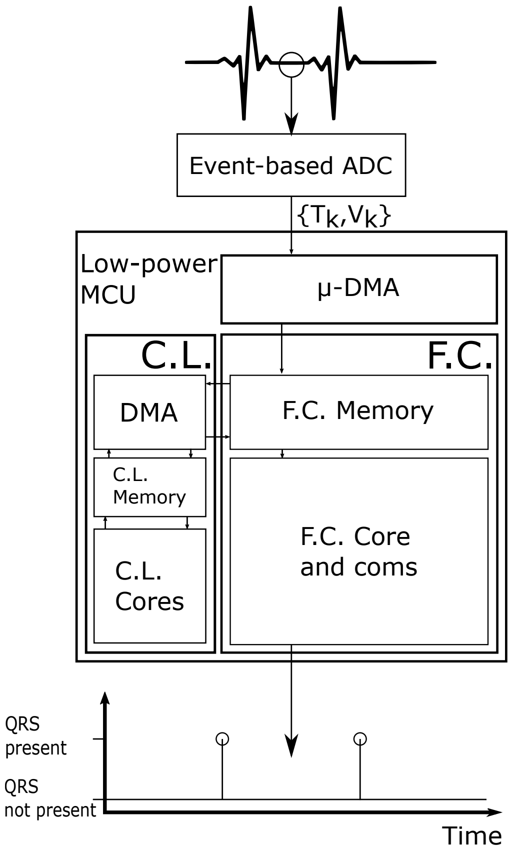

Fig. 2. Schematic representation of the whole system.

A. Event-based sampling

The central idea behind our approach is to improve the energy consumption for QRS complex detection by drastically reducing the amount of used data. In order to do so we move from the classical fixed-rate sampling to an event-based sampling where every data sample is acquired based on the occurrence of an event [1], [6]. In this work, we focused on the level-crossing event-based sampling technique: given an input range ΔV and a target resolution of B bits, we obtain $L = 2^B$ levels with a separation between them of $\delta v = \frac{\Delta V}{L}$. As soon as the input signal crosses one of these levels, a new sample $\{v, t\}$ is obtained. This is illustrated in Fig. 3.

For implementing this ADC stage, we selected the circuit described in [7], which is ready to be deployed on actual hardware. This device does not output directly the $\{v, t\}$ pairs, but an event signal spike and a direction signal (sign of the derivative of the signal), which allows to directly reconstruct the complete samples. Even if it is not possible to define a uniform frequency for this type of signals we can, anyway, define an average frequency in order to have a rough idea on the data reduction factor. As Fig. 4 shows, when the number of bits used decreases the obtained signal gets more and more sparse, making this event-based sampling a natural low-pass filter for noise while still preserving the important features of the signal. One important assumption that was made by observing the results of the event-based sampling is that the

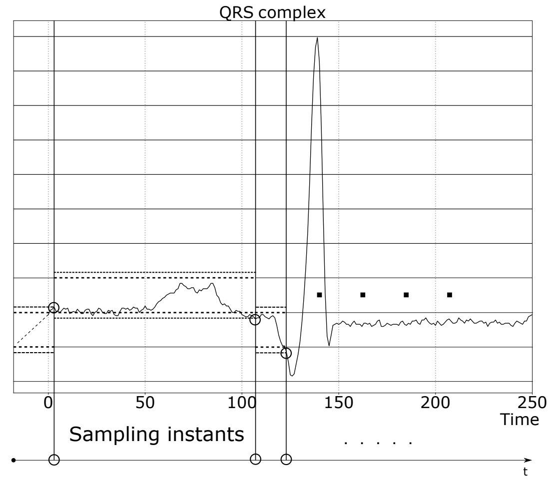

Fig. 3. Level crossing event-based sampling: in this example, the horizontal lines are the levels used, the thickest dotted line are the levels considered at that time instant and the smaller dashed lines represent the hysteresis around the levels considered. These hysteresis levels can be set by the user at any percentage of the δv and represent the actual value that the signal needs to cross to be sampled. [Source: Record 100 of the MIT-BIH Arrhythmia DB, lead MLII, between 01:12 and 01:13]

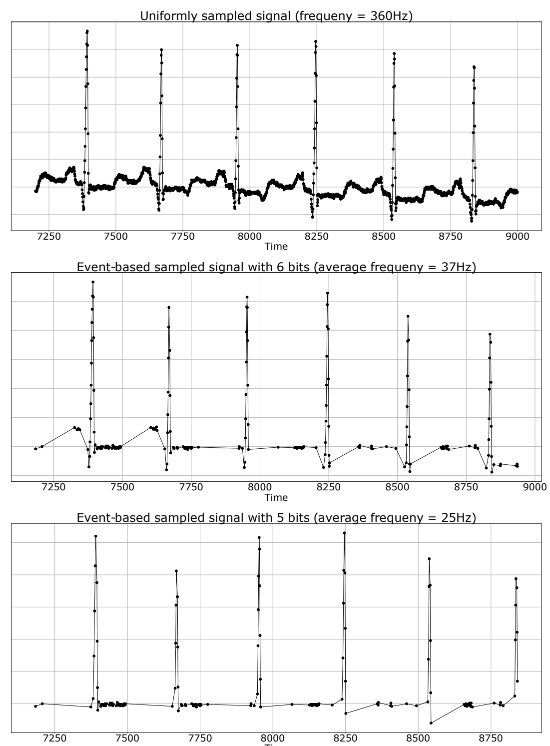

Fig. 4. Signal sampled with different event-based ADC resolutions. [Source: Record 100 of the MIT-BIH Arrhythmia DB, lead MLII, between 00:20 and 00:25]

distance between the true signal and its piece-wise linear approximation never grows too large. This is true once we fix a maximum Δt between the sampled points. Due to the discussed type of output of the ADC, a counter is needed to compute t for each sample. Using the overflow of this counter to signal that no level crossing happened in that time, this gives us the needed maximum time between samples and bounds the error between the linear interpolation and the true signal.

B. QRS detection algorithm

The main contribution of this work is a novel algorithm for the detection of QRS complexes on event-based sampled signals. Yet this algorithm has not been designed from scratch, but it is based on one of the most robust methods in the state of the art. The choice was based on two main reasons: the possibility of being re-adapted to a non-uniformly sampled signal, and the ability to design the full stack, from prototyping to implementation (in an emulator). This situation led to exclude all methods involving wavelet or Fourier transformations: even if a theory of wavelet on event-based sampled domain exists, the implementation would require more energy than the adaptation of classic algorithms, and a clear advantage in terms of detection performance has not been proven.

The considered algorithm is the `gQRS` algorithm, published in the WFDB software compilation from Physionet [8], and that is recognized as one of the best-performing algorithms on public ECG databases [9]. Other approaches exist for the detection of QRS complexes on event-based sampled signals [10], but they are fully coupled with the level-crossing sampling strategy. The hereby proposed method can work with any non-uniformly sampled signal that can be reconstructed by linear interpolation. In order to understand it better, we can divide the `gQRS` algorithm into three phases: filtering, integration and peak detection and thresholding.

1) Filtering: The first operation applied on the signal by the original `gQRS` algorithm is double filtering, that is illustrated in Fig. 5. Since all the used filters are digital, all of them operate with a delay time between their taps of $\delta t = \frac{1}{4} T_{QRS}$ where T_{QRS} is the average duration of the QRS complex. This is a settable parameter not learned during the execution, so we assume the default value of $T_{QRS} = 0.07s$. The first filter is a trapezoidal low-pass filter, which is used to smooth the signal and remove motion artifacts and electric noise. However, this filter was not implemented for two main reasons. First, as discussed above, the event-based sampling is already performing a low-pass filtering on the noise. Second, this filter is originally implemented with an infinite impulsive response (IIR) design. In order to apply an IIR filter in this environment we would have needed to re-sample its impulsive-response (IR) and approximating it with a Finite Impulsive Response (FIR) filter that would have added complexity (hence energy consumption) to the algorithm.

After this first filter, a second one is used to enhance the QRS peak in the smoothed signal and make the other parts of the signal lower. This is done using what the original algorithm calls an "adapted filter", whose IR shape is designed to resemble the shape of a typical QRS complex. Being the IR of this filter an odd function with respect to the central sample, this filter gives a strong response to both positive and negative QRS complexes. This filtering is obtained with a FIR digital filter that takes into account 9 samples at a distance of δt, introducing a delay on the filtered signal of $T_{QRS}(4\delta t)$.

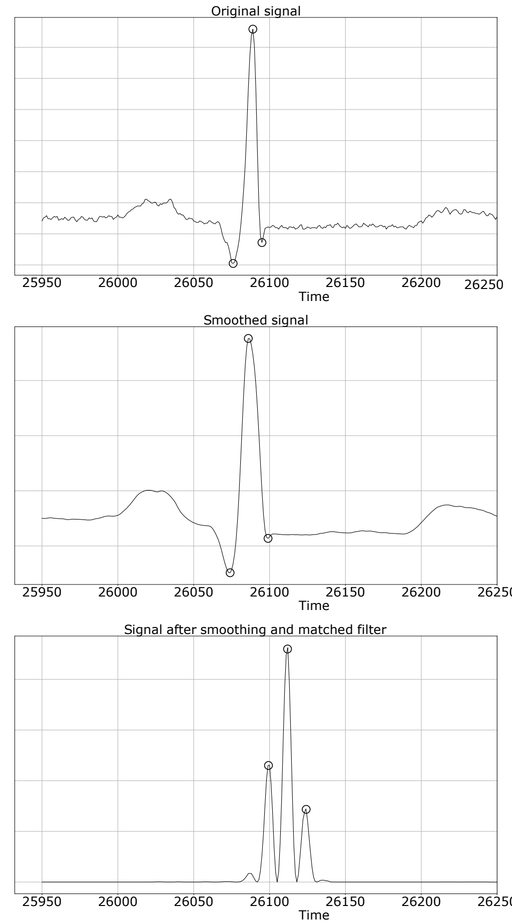

Fig. 5. Example of application of the double filtering. The first figure shows the raw data fed to the original `gQRS` algorithm, in the second figure we can see the effect of the smoothing filter on the raw data. Finally, in the third image, we can observe the results of the adapted filter. The circled points are the peak detected by the algorithm. Only the highest peak is classified as a QRS complex. [Source: Record 100 of the MIT-BIH Arrhythmia DB, lead MLII,between 01:12 and 01:13]

In the original `gQRS` algorithm, the results of this first stage get accumulated and then squared, performing a first order numerical integration. However, in our approach we exchanged these two steps, performing first the integration and then the filtering (and finally the squaring). This can be performed easily because the integral of the convolution is the convolution of the integral. We made this for a simple reason: our implementation relies on the idea that the error between the true signal and a piece-wise linear interpolation of it never grows too big. As we will see, this allows us to easily compute the integral between each sub-sampled point without the need of any re-sampling. Then, we apply the filter to each sub-sampled and integrated point by re-sampling only the strictly needed points on the integral signal. This means that if we want to apply the filter to the sub-sampled point at t_n we will need to just re-sample the four points before and the four points after it at a δt distance.

2) Integration: By computing the integral of the linear interpolation of the signal only for the obtained sub-sampled points, we can greatly reduce the computation workload and the memory needed to store the obtained values. Given two consecutive events at times $\{t_1, t_2\}$ and with values $\{v_1, v_2\}$, the following equation holds:

$$I_{t_1-t_2} = \sum_{i=t_1}^{t_2-1} \tilde{x}_n[i] = \sum_{i=t_1}^{t_2-1} (m \cdot i + q) \tag{1}$$

where $m = \frac{v_2-v_1}{t_2-t_1}$, $q = v_1 - m \cdot t_1$ and $\tilde{x}_n[i]$ is the linear interpolation between the points of the event-based sampled signal. Substituting $t_2 - 1$ with l and using the Gauss sum, we can write:

$$\begin{aligned} I_{t_1-t_2} =& m \cdot \sum_{i=t_1}^{l} m \cdot i + q \cdot (t_1 - l + 1) = \\ & \frac{m}{2} \cdot (l + t_1)(l - t_1 + 1) + q \cdot (l - t_1 + 1) = \\ & \frac{m}{2} \cdot (t_2(t_2+1) - t_1(t_1+1)) + q \cdot (t_2 - t_1) \end{aligned} \tag{2}$$

This gives us the solution in closed form of the numeric integral using only the points present in the event-based signal. Thus, writing m and q in function of $\{v_1, v_2\}$ and $\{t_1, t_2\}$, even if possible, does not bring any advantage. Moreover, by storing those values (m and q), we do the work needed for re-sampling and filtering in a slightly more efficient way.

3) Peak detection and thresholding: Finally, the third step of the algorithm checks the presence or absence of a peak and determines whether it is from a QRS complex. This operation can divided into four steps:

a) We first check if a peak structure is present, verifying if $x_{t-1}^2 < x_t^2 > x_{t+1}^2$. If such structure is present, we compare the middle value (cuspid of the peak) with a first threshold to reject peaks caused by noisy signals. Peaks that pass this first step are saved in a buffer of peaks.

b) We analyze all the peaks inside a certain time window determined by physiological constraints, affecting the maximum and minimum separation between actual QRS complexes. For every peak in this window, we check if it is dominant (i.e. the highest) in a direct neighbourhood of peaks.

c) For all the dominant peaks, we compare them with a second adaptive threshold that fits the local properties of the detected QRS complexes.

d) Finally, all the discarded dominant peaks are analyzed backwards in order to recover any possible missed QRS peaks.

C. Ultra-low power implementation

The targeted MCU used is the open-source PULP platform [11], and more specifically the version called "Mr. Wolf" [12]. It is an advanced microcontroller based on an ultra-low-power 32-bit RISC-V processor and with an embedded 8-core cluster for parallel computing. We can ideally split the chip in two main domains: the Fabric Controller (FC) and the Cluster (CL). These domains are defined by zones under the control of the main core or of the cluster and by separated power management.

In order to be able to parallelize the `gQRS` algorithm a re-design of its central part was needed. While it was possible to parallelize integration and filtering, the thresholding step adapts the thresholds based on previous results and it was not possible to use the same techniques used to parallelize the integration. Not only the structure of the parallelized code was changed, but we also needed an efficient method for data transfer between the FC and the CL. We will describe now these three main steps, and a flow diagram of the code is shown in Fig. 7. First, the MCU acquires in sleep mode a predefined-length buffer of data. Then the buffer is sent to the CL that will execute the integration and filtering. Finally, the results are handled back to the FC that will perform the detection of QRS complexes.

FC-CL Bridge: The data transfer between the FC and the CL is illustrated in Fig. 6. In order to send data to the CL, the FC acquires two data buffers (vectors) composed of head, body and tail: time ($\vec{t_k}$) and value ($\vec{v}_k$) where k is the k^{th} buffer of data. The buffer length can be set-up by the user in order to adapt to different scenarios: each buffer needs to be filled for an average time of $\overline{T} = \frac{v_{len}}{\overline{f_s}}$ where $\overline{f_s}$ is the average sampling frequency and v_{len} is the length of the body of the buffer. Therefore, given a target $\overline{T}$, we can adjust v_{len} based on the resolution of the ADC (that determines $\overline{f_s}$). In this work we used $v_{len} = 460$ for a $\overline{f_s} = 23\ Hz$, reaching a latency time between executions of 20 seconds.

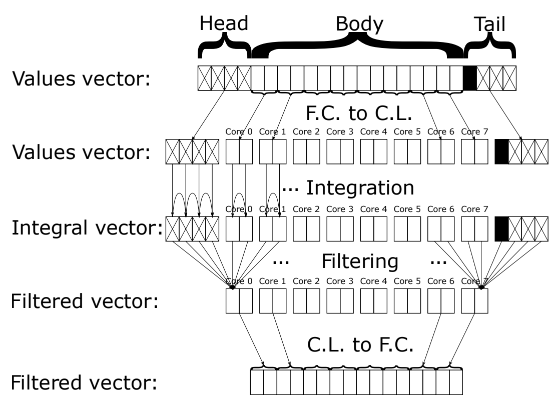

Fig. 6. Example of the vector handling (values) for the Multi-Core (Wearable) implementation. We can observe that from the first element of the tail (black) all future values get ignored. Because of this, the body of the next vector needs to start with the first element of the tail.

The tail and the head are needed to apply the matched filter to the first and last elements of the body without the need to add padding and delete a certain amount of values later. Once the two buffers are filled, the MCU wakes up and the CL performs a Direct Memory Access (DMA) transfer in order to bring the data from the FC memory and starts to operate on them. Once the CL has finished, it puts the results of the integration and filtering in a return vector $\vec{r}_k$ and starts a second DMA transfer that copies the result vector to the FC memory. Once this copy is finished, the FC starts the peak detection and thresholding procedure. Once $\vec{r}_k$ has been completely analyzed, the FC copies the last elements of $\vec{t}_k$ and $\vec{v}_k$ at the beginning of $\vec{t}_{k+1}$ and $\vec{v}_{k+1}$ in a way that the next first element of the body will be the previous first element of the tail (black square in Fig. 6).

Integration: As $\vec{t}_k$ and $\vec{v}_k$ arrive, we divide the body, the tail and the head in eight contiguous parts (chunks) and send them to the eight cores of the cluster. Each core applies the integration in Eq. 2 to the chunk received and puts the results in three vectors (The vectors of the I, m and q) shared between the cores. Note that this vector is not the total integral vector. It is a vector composed of the integral of the chunks.

Filtering: Each integrated chunk is then filtered. In order to implement the filtering around a central point, we need other eight equally spaced points at a distance of δt. To obtain them, we must use the re-sampling technique already discussed but with some minor changes. The integral vector used will be discontinuous from chunk to chunk. To deal with this, whenever the adapted filter is applied to a point that is in the i^{th} chunk, we calculate the required timing of each interpolated point, knowing that every needed point will fall between 2 points already present. For every needed point we have three conditions to check, namely:

1) The point is in the $i^{th}-1$ chunk. In this case, we simply implement the discussed re-sampling algorithm
2) The point is in the i^{th} chunk or between the i^{th} and the $i^{th}-1$ chunks. In this case, we sum the last value of the $i^{th}-1$ chunk to the two points that we will use for the interpolation.
3) The point is in the $i^{th}+1$ chunk or between the $i^{th}+1$ and the i^{th} chunks. In this case we sum the last value of the $i^{th}-1$ chunk and the last value of the i^{th} chunk to the two points that we will use for the interpolation.

After this check, we can apply the interpolation formula and complete the filtering.

III. Experimental Results

The experimental evaluation of this work has focused on two different aspects: the detection performance of the developed algorithm and its energy consumption. The data used for this work were acquired by departing from an already existing database: the MIT-BIH Arrhythmia Database [13]. This is a public database composed of 48 recordings of 30 minutes duration and with all beats manually labeled. The recordings were acquired at a sampling frequency of $F_s = 360Hz$ per channel with 11-bits resolution over a 10 mV range with 2+1(reference) leads placed on the chest with the exception of one recording that was registered using only 1+1 leads. In this work, we will use only the signal coming from the MLII lead, since it is the most common one. However, in records 102 and 104 this lead is not present, so they were discarded. To obtain the event-based sampled data from this database, we used a software emulator of the selected level-crossing ADC [7].

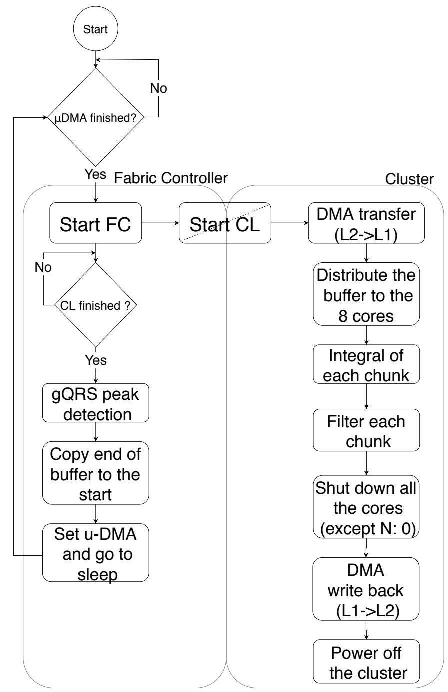

Fig. 7. Block design for the multi-core version of the `gQRS` algorithm

A. QRS detection performance

The detection performance of the algorithm has been assessed using the *F1-Score* metric, which is defined as the harmonic mean between the *Specificity* and *Positive-predictivity*, according to the following formulae:

$$S_p = \frac{\textit{Correct QRS predicted}}{\textit{Total number of true QRS peaks}} \tag{3}$$

$$P_p = \frac{\textit{Correct QRS predicted}}{\textit{Total number of QRS predicted}} \tag{4}$$

$$F_1 = \frac{2 * S_p * P_p}{S_p + P_p} \tag{5}$$

We compared the F1 score of the adapted `gQRS` (that works on event-based sampled signals) with the original `gQRS` algorithm (that works only with uniformly-sampled signals). We linearly re-sampled the event-based signal in order to be able to apply the original `gQRS` algorithm to this type of signal. As shown in Fig. 8, the performance degradation between the adapted version of the `gQRS` algorithm and its original implementation is marginal for the highest resolutions. Also considering Fig. 8 and the average sampling frequencies shown in Table I, we can conclude that the optimal ADC resolution in this case is 5 bits. If we move to 6 bits, the average sampling frequency increases 1.7 times, while the performance improvement is minimal. On the other side, the high penalty in performance when we move to 4 bits would make the algorithm unusable in practice.

TABLE I
AVERAGE FREQUENCIES (ALONG ALL THE DATA-SET) FOR THE EVENT-BASED SAMPLED SIGNALS USING DIFFERENT ADC RESOLUTIONS

	4 Bits	5 Bits	6 Bits	7 Bits
Average sampling frequency [Hz]	11.8	26.2	45.8	75.0

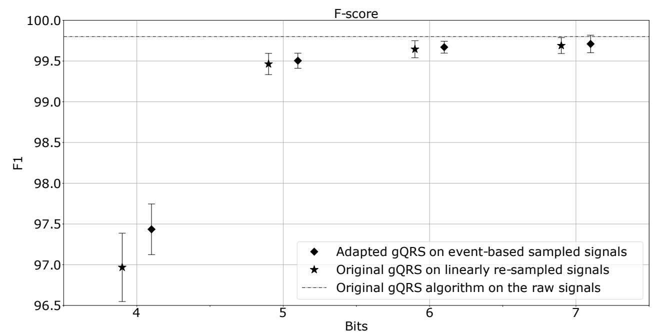

Fig. 8. F1 score (calculated globally with all the records in the database) comparison between the adapted `gQRS` algorithm and the original one using different numbers of bits for the event-based sampling. The vertical bars represent $\pm 1\sigma$ (median absolute deviation).

B. Energy consumption

In order to evaluate the energy consumption of the adapted `gQRS` version, we implemented both the original `gQRS` algorithm and the proposed one on the selected MCU, as discussed in Section II-C. The results can be divided into two sections, which are shown in Table II: Energy in Run and Energy in Sleep. They were obtained by measuring the energy consumption in a window of 20 seconds of data. The original algorithm works with the original signal sampled at 360 Hz, while the event-based version used the signal sub-sampled with 5 bits, giving an average frequency of 23 Hz.

1) Energy in Run: The energy in run is the energy used strictly for the execution of the two algorithms with the MCU in run state. We can observe that the developed version of the `gQRS` algorithm results to be 15.6 times more energy-efficient than the original one.

2) Energy in Sleep: The energy consumption in sleep mode is due to processor and memory leakage while the MCU is acquiring data. Because even the original `gQRS` is already highly efficient in terms of computation, the total time in sleep remains almost constant among the two different implementations. In particular, the measurements obtained in Table II come from the processing of 20 seconds of data and the sleep time ranges from 19.935 seconds to 19.998 seconds. In this context, the total energy consumption is dominated by the leakage current needed for memory retention while the platform is in sleep. To improve this aspect, we envision two complementary strategies: 1) a tailored memory hierarchy that shrinks the minimum block size from the current 64 KB to 1 KB, reducing the leakage proportionally since only one block would be required by `gQRS`; and 2) scaling the used technology, moving from 40 nm to 22 nm, which gives a quadratic leakage reduction in the best case. By combining both strategies, the potential energy reduction factor in sleep mode would be of $\frac{64}{1} \cdot \frac{40}{22}^2 \approx 212$.

TABLE II
ENERGY USAGE OF THE TWO DIFFERENT VERSIONS OF `gQRS` FOR THE PROCESSING OF 20 SECONDS OF DATA IN THE ULTRA-LOW POWER MCU. E_r: energy in run, E_s: energy used during sleep, T_r: processing time, E_{tot}: total energy used for a 20-second window.

	E_r [mJ]	E_s [mJ]	T_r [ms]	E_{tot} [mJ]
Original `gQRS`	0.2534	1.53	44.6	1.78
Event-based `gQRS`	0.0162	1.53	1.85	1.55

IV. CONCLUSION AND FUTURE WORK

In this work we explored the possibility of using event-based sampling to reduce the amount of used energy for data processing in biosignal monitoring devices. In particular, we focused on the task of QRS detection in ECG signals, adapting the `gQRS` algorithm in order to make it work in this environment. Then we implemented it on an ultra-low power, multi-core MCU. When compared to the original `gQRS` algorithm on a uniformly-sampled signal at 360Hz, our implementation achieves an increase in energy efficiency up to fifteen times without a noticeable performance degradation. The full source code of the algorithms and the experiments has been published under an open-source license for reproducibility[1]. As a future work, the extremely small running time of QRS detection with this proposed approach gives us the opportunity to go further in the performed analysis of the signal, including more complex tasks such as wave delineation, heart beat classification or even rhythm and respiration analysis.

REFERENCES

[1] M. Miskowicz, *Event-Based Control and Signal Processing*. Embedded Systems, CRC Press, 2017.

[2] D. Craven, B. McGinley, L. Kilmartin, M. Glavin, and E. Jones, "Compressed Sensing for Bioelectric Signals: A Review," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, pp. 529–540, mar 2015.

[3] M. Zaare, H. Sepehrian, and M. Maymandi-Nejad, "A new non-uniform adaptive-sampling successive approximation ADC for biomedical sparse signals," *Analog Integrated Circuits and Signal Processing*, vol. 74, pp. 317–330, feb 2013.

[4] L. Sörnmo and P. Laguna, "Chapter 6 - The Electrocardiogram: A Brief Background," in *Bioelectrical Signal Processing in Cardiac and Neurological Applications*, pp. 411–452, Academic Press, 2005.

[5] L. Sörnmo and P. Laguna, "Chapter 7 - ECG Signal Processing," in *Bioelectrical Signal Processing in Cardiac and Neurological Applications* (L. Sörnmo and P. Laguna, eds.), pp. 453–566, Academic Press, 2005.

[6] Z. Tian, R. Ying, P. Liu, G. Wang, and Y. Lian, "Event-driven analog-to-digital converter for ultra low power wearable wireless biomedical sensors," in *2015 IEEE 11th International Conference on ASIC (ASICON)*, pp. 1–4, IEEE, nov 2015.

[7] G. Rovere, S. Fateh, and L. Benini, "A 2.2-μw cognitive always-on wake-up circuit for event-driven duty-cycling of iot sensor nodes," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, pp. 543–554, Sep. 2018.

[8] A. L. Goldberger *et al.*, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals," *Circulation*, vol. 101, pp. 215–220, June 2000.

[9] M. Llamedo and J. P. Martínez, "QRS detectors performance comparison in public databases," in *Computing in Cardiology 2014*, pp. 357–360, Sep. 2014.

[1] https://c4science.ch/source/Event_based_gQRS/

[10] S. A. H. Sabzevari, N. Ravanshad, and H. Rezaee-Dehsorkh, "An Ultra-Low-Power QRS-Detection System Based on Level-Crossing Sampling," *Iranian Conference on Electrical Engineering (ICEE)*, pp. 1456–1461, 2018.

[11] E. Zürich and U. of Bologna, "PULP-Platform." https://www.pulp-platform.org/, 2017. [Online; accessed 20/06/2019].

[12] E. Zürich and U. of Bologna, "Mr. Wolf details." http://asic.ethz.ch/2017/Mr.Wolf.html, 2017. [Online; accessed 20/06/2019].

[13] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH arrhythmia database," *IEEE EMBM*, vol. 20, no. 3, pp. 45–50, 2001.