Wireless EEG System Achieving High Throughput and Reduced Energy Consumption Through Lossless and Near-Lossless Compression

Guillermo Dufort y Álvarez, Federico Favaro, Federico Lecumberry, Álvaro Martín, Juan P. Oliver, Julián Oreggioni, Ignacio Ramírez, Gadiel Seroussi and Leonardo Steinfeld.

***Abstract*—This work presents a wireless multi-channel EEG recording system featuring lossless and near-lossless compression of the digitized EEG signal. Two novel, low-complexity, efficient compression algorithms were developed and tested in a low-power platform. The algorithms were tested on six public EEG databases comparing favorably with the best compression rates reported up to date in the literature. In its lossless mode, the platform is capable of encoding and transmitting 59-channel EEG signals, sampled at 500 Hz and 16 bits per sample, at a current consumption of 337 μA per channel; this comes with a guarantee that the decompressed signal is identical to the sampled one. The near-lossless mode allows for significant energy savings and/or higher throughputs in exchange for a small guaranteed maximum per-sample distortion in the recovered signal. Finally, we address the trade-off between computation cost and transmission savings by evaluating three alternatives: sending raw data, or encoding with one of two compression algorithms that differ in complexity and compression performance. We observe that the higher the throughput (number of channels and sampling rate) the larger the benefits obtained from compression.**

I. Introduction

MONITORING brain activity can play an important role in understanding the functioning of the human brain, as well as in potentially improving our quality of life [2]. The electroencephalogram (EEG) is one of the main tools used for studying brain activity. However, current standard EEG systems are wired and uncomfortable, and are mainly used in static settings in clinical practice. In order to enable EEG recordings in daily-life activities, EEG technology needs to become wearable (wireless, low weight, and small size), which requires low-power operation and energy-efficient wireless data transmission.

Although a bandwidth ranging from 0.5 Hz to 60 Hz is sufficient for many EEG applications [3], much higher frequencies (up to 500 Hz) are required in other cases [2], [3]. In addition, the current miniaturization of analog front-ends (AFE) for acquiring EEG signals enables the simultaneous recording of hundreds of channels [4], [5]. As a consequence, handling high data rates efficiently is essential for high-performance EEG recorders.

This work is a substantially extended and improved version of [1]. It was partially funded by CSIC-UDELAR (Comisión Sectorial de Investigación Científica, Universidad de la República, Uruguay), ANII (Agencia Nacional de Investigación e Innovación, Uruguay) and CAP-UDELAR (Comisión Académica de Posgrado, Universidad de la República, Uruguay).

The authors are with Universidad de la República, Montevideo, Uruguay. Gadiel Seroussi is also with XPERI Corp., CA, USA.

In this scenario, data compression becomes a key factor in a wireless EEG platform, not only for reducing power consumption (usually driven by the transmission), but also to overcome wireless technology limitations [6]. For example, a system with 64 channels, 16-bits per sample, at 1 kilo-samples per second (ksps), requires a payload data rate of 1 Mbps, which is a throughput attainable by Bluetooth but not by other low-power transmission protocols such as IEEE 802.15.4. Moreover, common low-power transmission protocols available at this moment are unable to support 256 channels (payload data rate of 4 Mbps).

EEG data acquired for clinical purposes is often required to be processed, transmitted and stored without distortion; this establishes the need for lossless compression algorithms, in which the decompressed digital signal is identical to the originally captured one. If the preceding requirement is relaxed to allow a small, prescribed maximum per-sample distortion on the recovered signal, we arrive at the so called *near-lossless* setting. The near-lossless setting allows for significantly higher data rates and/or number of channels, with a user-controlled maximum sample reconstruction error given by a parameter δ. This configuration guarantees that the reconstructed value of each sample differs by up to δ quantization levels from the originally acquired sample. Several lossless, near-lossless and lossy methods (the latter case refers to distortion levels that are only guaranteed on average, in contrast to per-sample) have been proposed during the past 20 years for EEG and other biomedical signals [6]–[12]. We review the most relevant literature in Section III.

The impact of EEG compression on the overall energy consumption of an electroencephalograph is driven by two factors that are generally opposed: the better the compression ratio, the more energy saved on transmission, but the more complex the compression algorithm, the greater the energy consumed in computing. In this paper we present two novel low-complexity EEG compression algorithms and evaluate this trade-off in actual hardware. To this end we define in Section II a low-power platform suitable for wireless EEG where we implement the compressors. The algorithms, which are inspired on the same statistical model as [10], are presented in Section III. Both admit lossless and near-lossless variants, and require only basic operations, which can be readily implemented using

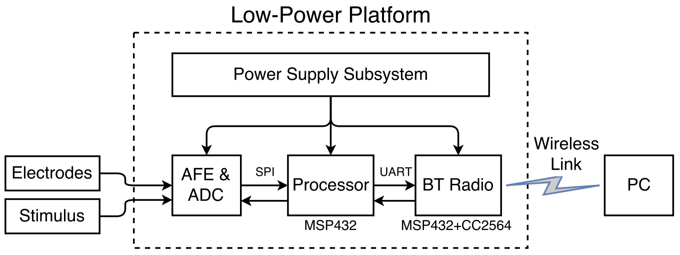

Fig. 1. Wireless EEG recording system block diagram.

discrete logic blocks as part of a custom System on Chip (SoC) such as the ones described in [13], [14].

We analyze the aforementioned trade-off by means of simple models for the power consumption invested in data compression and that spent in data transmission, which allows for extrapolating the system performance to different hardware platforms. This analysis and experimental evaluation of the models is presented in Section IV. In Section V, we experimentally evaluate the compression performance of the proposed platform, and we report on the power consumption obtained with different compression alternatives and different configurations of sampling rate and number of channels. The results show that, for high throughput settings, the compression algorithms yield significant power savings. For example, the transmission of a raw 59-channel EEG signal sampled at 500 Hz and 16 bits per sample resolution over a Bluetooth link consumes approximately 28.2mA, while compressing and transmitting the same signal consumes about 2mA less. Moreover, the reduction in the transmitted data rate due to compression allows for the use of the so-called *sniff* mode of Bluetooth making the current consumption drop to 19.9mA (sniff mode falls short of bandwidth to transmit the raw signal). In addition to power saving we also evaluate the compression performance of the two proposed algorithms on public EEG databases, obtaining results that are competitive with state of the art EEG compressors. Conclusions are presented in Section VI.

II. Low-power platform

The EEG platform, depicted in figures 1 and 2, comprises an analog front-end (AFE), an analog to digital converter (ADC), a low-power processor, a Bluetooth (BT) radio transceiver and a power supply subsystem. All modules are powered by a 3.3V dc source. The analog EEG signal is acquired from a set of electrodes and fed into the AFE.

The AFE and ADC stages in our platform comprise two off-the-shelf RHD2132 chips from Intan Technologies. Each RHD2132 chip is able to acquire, amplify, digitize and transmit via a Serial Peripheral Interface (SPI) up to 32 channels at 30 ksps each. The RHD2132 chip features low input referred noise ($2.4\mu V_{\rm rms}$[1].), programmable bandwidth and low power operation. For instance, the total current consumption of the two chips to acquire 64 channels at 500 sps/ch is 1.8 mA and at 1 ksps/ch it is 2.1 mA.

[1]The thermal noise level is less than 200 $nV/\sqrt{Hz}$ with a $1/f$ noise corner of 2.3 Hz [15]

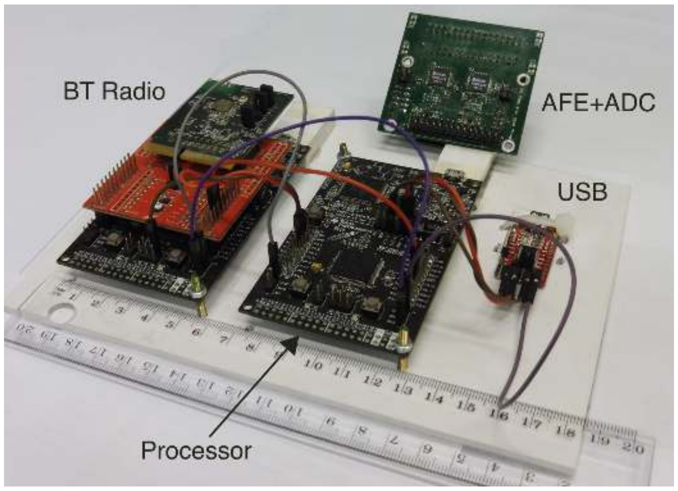

Fig. 2. Low-Energy High-Throughput EEG Wireless System.

The processor block consists of a Texas Instruments MSP432P401R microcontroller, a 32-bit ARM Cortex-M4F microcontroller with a maximum clock frequency of 48 MHz, with 256 kB of Flash and 64 kB of RAM memory. This chip features a typical power consumption of 4.6 mA in *active mode* and offers severals modes of low-power operation, called *sleep mode*, where its power consumption can be as low as hundreds of nanoamperes. In addition, this microcontroller includes a rich set of peripherals including the SPI serial port used in our platform to communicate with both RHD2132 chips, the UART serial port to communicate with the BT radio and a timer to control the sampling frequency.

The BT radio transceiver core is a module based on a CC2564 chip by Texas Instruments. This is a dual mode module that supports Bluetooth 4.1 in low energy mode (BLE) and basic (BR) or enhanced data rate (EDR) mode. Our prototype uses the EDR mode with serial port profile (SPP); this allows for high throughput configurations, e.g. 31 channels at 1 ksps/ch, which would not be affordable with BLE 4.1.

The processor embedded software is responsible for receiving the sample data, running the compression algorithm, and transmitting the compressor output to the BT module. A round-robin with interrupts architecture is adopted, where interrupt service routines (ISR) are extensively used to exchange (transmit and receive) data, and keep the processor in sleep mode while no processing is needed. The microcontroller's timer is used to trigger a new sample acquisition. The samples (one from each channel), received via the SPI interface, are stored in a input buffer. Once the input samples of all channels are received, the compression algorithm is executed. The compressor output is stored in an output buffer to be transferred to the BT module through the UART interface. Once completed, the microcontroller enters in sleep mode.

In order to asses our platform using a controlled setup, the software module responsible for receiving the sample data from the RHD2132 chips via SPI is replaced by a Test Double. The Test Double module supplies data that is either received via a USB interface from a PC (Section IV-A) or read directly from the processor memory (Section V).

III. EEG Compression algorithms

The lossless, real time and low power requirements of our platform impose severe restrictions on the latency and computational resources of its embedded software.

To start, the real time requirement rules out any method that requires two or more passes over the whole dataset. Other methods perform two or more passes on blocks of data. If B is the length of the block and f_s is the sampling frequency, this results in a lag of B/f_s seconds. This is the case of the MPEG-4 audio lossless coding standard [16] (ALS), which has also been applied to biomedical signal compression [17]. Unfortunately, the block sizes required for ALS to be effective (above 2048 samples) result in lag of several seconds for typical EEG applications. This is also the case of transform-based methods such as [18]–[20], which use different kinds of linear transforms to remove correlation both spatially (between different electrodes across the scalp) and temporally (between samples at different sampling times).

In the case of transforms, there are additional computational issues. First, the number of operations per sample scales superlinearly with the number of channels C and the length of the block B. This is at least proportional to $C \log C$ for Fast Wavelet or Fourier transforms applied only to inter-channel decorrelation, $(BC)\log(BC)$ when such transforms are applied to multi-channel blocks as in [19], and as high as $(BC)^3$ (the cost of performing a Singular Value Decomposition) for adaptive transform methods such as [20]. A higher number of operations translates directly into a higher power consumption, rendering the aforementioned methods unsuitable for low-power applications. Also, for real-time transmission, the allowable computational complexity cannot exceed the maximum number of operations that can be performed by the hardware within a sampling period $T_s = 1/f_s$. Moreover, transform methods require the complete block to be stored in memory, thus imposing higher memory requirements on the hardware; even in-place integer-based transforms would normally require at least $2BC$ bytes of buffer size for 16 bit samples (for minimum lag, a double-buffer strategy should be used, thus doubling that number).

Finally, methods such as [19]–[22] use an *arithmetic coder* [23] which is significantly more computationally demanding than more specialized ones such as the Golomb-Rice coder [24].

At the time of this writing and to the best of our knowledge, the method that offers the best compression ratio reported in the literature is the algorithm described in [10]. This is a low-latency, low-complexity algorithm (the complexity actually grows linearly in storage and number of operations with respect to the number of channels), with controllable per-sample distortion. Thus, we choose [10] as our starting point; the algorithms developed hereafter in this work involve non-trivial modifications of this method with the goal of making it suitable for implementation on a low-power microcontroller with minimum computational and memory requirements. In Section V we discuss why [10] is not directly applicable in such environment.

As most EEG compression algorithms, the method in [10] exploits temporal and spatial sample correlations. These are induced by natural properties of the target signal such as temporal continuity, natural correlation of neural activity across regions, and spatial smoothing due to the different layers of tissue that separate the source signals (the neurons) from the point where they are measured (the electrodes).

The essence of the algorithm (see Figure 3) is summarized below (see [10] for further details). Later on, we elaborate on the components of the algorithm that have been modified significantly.

- The coding stage is predictive: both encoder and decoder predict the value of each sample from previously encoded samples; the actual value is described to the decoder by encoding the difference with respect to the prediction using the *Golomb-Rice code* (see, e.g., [25]).
- Channel samples are encoded in a pre-specified order following a tree; the root channel is predicted using past samples only, whereas all other channels have a *parent channel* (corresponding to their parent in the tree) that "helps" them, meaning that the past (and present) information about the parent channel is used for predicting the present sample of the child channel.
- Each sample prediction is a weighted average of a set of linear predictions of different orders, which are combined using an *exponential weighting* [26] scheme to form a final prediction.
- All these linear predictions are adaptive; they are updated in an online fashion using an efficient implementation of a multi-channel Recursive Least Squares (RLS) algorithm [27].

The performance and memory constraints of the target platform make the RLS algorithm used in [10] infeasible for high throughput scenarios and, in general, not very competitive in energy consumption (see discussion in Section V). Instead, we use a multi-channel extension of a simple integer-based, adaptive, single-channel prediction algorithm originally proposed by Speck in [28]. This extension, detailed in Subsection III-A, is an original contribution of this work. It turns out that although a compressor implemented with this predictor is significantly less complex, and thus requires a fraction of the resources, it still attains a performance similar to that of a full-fledged floating-point RLS implementation (see Table II in Section V-B). As an additional contribution, we propose, in Subsection III-B, an efficient integer implementation of the exponential weighting algorithm, which further improves the performance of the predictor. These tools, together with a cautious selection of a reduced set of predictors and other computation savings described in Subsection III-C, result in a very simple and efficient compression algorithm that we refer to as MCS (Multi-Channel Speck). In Subsection III-D, by replacing the adaptive predictors by fixed ones, we derive a significantly faster algorithm at the cost of some compression performance degradation, termed MCF (Multi-Channel Fixed). In Subsection III-E we describe a near-lossless encoding scheme that applies to both MCS and MCF.

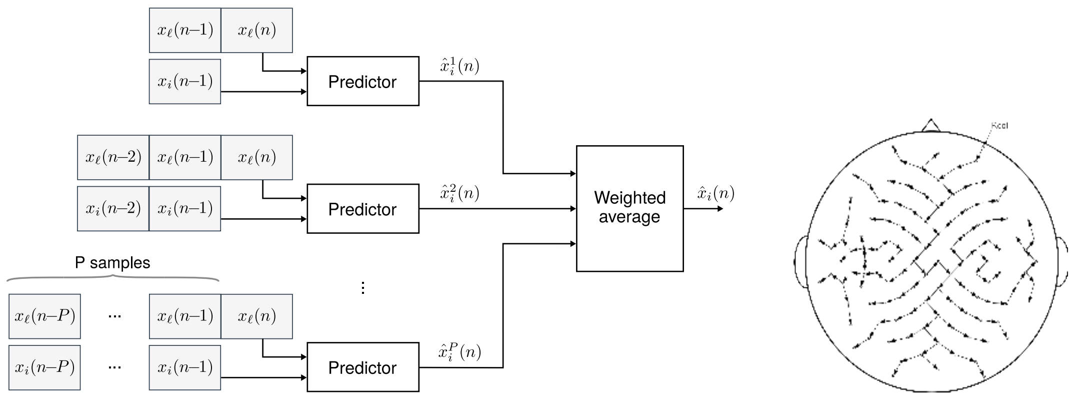

Fig. 3. Compression algorithm of [10]. LEFT: block diagram of the prediction scheme; here $x_i(n)$ refers to the value of channel i at discrete time n, x_ℓ is the "helper" (parent) channel of x_i, P is the maximum order of the predictors, $\hat{x}_i^p$ is the p-th order prediction of x_i and $\hat{x}_i$ is the final prediction for that channel. RIGHT: sample tree used when deciding which channel helps which; the root channel is encoded with no help.

A. Multi-Channel extension of the Speck algorithm

We consider a discrete time m-channel signal, $m > 1$. We denote by $x_i(n)$ the $i-$th channel (scalar) sample at time instant n, $n \geq 1$, and we refer to the vector $(x_1(n), \ldots, x_m(n))$ as the *vector sample* at time instant n. We assume that all scalar samples are quantized to integer values in a finite interval $\mathcal{X}$.

A general linear predictor of order p for a sample $x_i(n)$ as a function of the past samples of the same channel i, $x_i(1) \ldots x_i(n-1)$, is defined as

$$\hat{x}_i^p(n) = \sum_{k=1}^{p} a_{i,k} x_i(n-k)\,, \tag{1}$$

where $(a_{i,k} : k = 1, \ldots, p)$ are real coefficients. The Speck algorithm[2] defines $a_{i,k} = \hat{a}_{i,k}/K$ as a rational number, where $(\hat{a}_{i,k} : k = 1, \ldots, p)$ are integer coefficients and K is an integer normalization constant (usually a power of two, so that division by K can be carried out using bitwise-operators). The coefficients $\hat{a}_{i,k}$ are sequentially adapted upon comparing the prediction $\hat{x}_i^p(n)$ with the actual sample $x_i(n)$; we omit the dependence of $\hat{a}_{i,k}$ and $a_{i,k}$ on $x_i(1) \ldots x_i(n-1)$ for the sake of notation relief. The coefficient initialization and adaptation steps in the original single-channel scheme of [28] are specified next.

- *Initialization:* The coefficients $\hat{a}_{i,k}$ are initialized as
$$\hat{a}_{i,k} = K\,/\,p + \begin{cases} 1, & k \leq K\,\%\,p\,, \\ 0, & \text{otherwise}\,, \end{cases} \tag{2}$$
where K/p and $K\%p$ denote integer quotient and remainder, respectively.
- *Adaptation:* Let $\epsilon_i(n) = x_i(n) - \hat{x}_i(n)$ be the *prediction error* at time n, and $\operatorname{sgn}(\epsilon_i(n))$ its sign. If $\epsilon_i(n) = 0$, no adaptation takes place; otherwise, the coefficients $\hat{a}_{i,k}$ associated to the largest and smallest (signed) past p samples $(x_i(n-k) : k = 1, \ldots, p)$ are respectively decreased and increased by $\operatorname{sgn}(\epsilon_i(n))$; ties are broken by some fixed policy, e.g., choosing the coefficients with smallest index.

[2]The definition in [28] applies to digital images; our description is a straightforward adaptation to one-dimensional signals.

The preceding initialization and update procedures ensure that the coefficients $a_{i,k}$ add up to unity for all i; notice, however, that some of them may become negative.

In the scheme proposed in [10], the prediction $\hat{x}_i^p(n)$ for channel i depends on the p most recent samples of channel i, the p most recent samples of its *parent* or *helper* channel ℓ, and the *current* sample of channel ℓ, which is encoded before $x_i(n)$. Thus, we have

$$\hat{x}_i^p(n) = \sum_{k=1}^{p} a_{i,k} x_i(n-k) + \sum_{k=0}^{p} b_{i,k} x_\ell(n-k), \tag{3}$$

where $a_{i,k}$ and $b_{i,k}$ are (adaptive) real coefficients.

A straightforward extension of the Speck algorithm could be defined by applying its initialization and update procedures to the concatenation of $(\hat{a}_{i,k} : k = 1, \ldots, p)$ and $(\hat{b}_{i,k} : k = 0, \ldots, p)$ in terms of the concatenation of samples from both channels, $(x_i(n-k) : k = 1, \ldots, p)$ and $(x_\ell(n-k) : k = 0, \ldots, p)$. However, we have observed that this direct extension results in a poor performance when the mean values of channels i and ℓ differ significantly.

Instead, we apply it to centered versions of the channels, which we obtain by subtracting from each channel an on-line estimation of its mean, $\bar{x}_i(n)$, given by,

$$\bar{x}_i(n) = (1-\beta)\bar{x}_i(n-1) + \beta x_i(n), \tag{4}$$

where $0 < \beta < 1$ is a parameter. Although there is no known theoretical prediction performance guarantee (not even for the single channel Speck predictor), we have obtained very good results in practice (see Section V-B).

To implement (4) using integer-only arithmetic, we define an auxiliary variable $s_i(n) \triangleq \beta^{-1}\bar{x}_i(n)$ and rewrite (4) as

$$\begin{aligned} s_i(n) &= \beta^{-1}\bar{x}_i(n-1) - \bar{x}_i(n-1) + x_i(n) \\ &= s_i(n-1) - \bar{x}_i(n-1) + x_i(n). \end{aligned}$$

Now the recursion is expressed only in terms of additions and subtractions. By choosing β to be a negative integer power of two, $\beta = 2^{-b}$, we get $\bar{x}_i(n) = s_i(n) \gg b$ (where $\gg$ denotes a bitwise arithmetic shift-right operation).

B. Fast exponential weighting

Exponential weighting, a key feature in the predictive performance of our algorithm, is a well-studied method with a solid theoretical justification [26]. In this scheme, the final prediction of a sample is a weighted average of the outputs of a set $\mathcal{P}$ of predictors working in parallel,

$$\hat{x}_i(n) = \frac{\sum_{r\in\mathcal{P}} w_r(n)\hat{x}_i^r(n)}{\sum_{r\in\mathcal{P}} w_r(n)} , \tag{5}$$

where $w_r(n)$ is a positive weight that decays exponentially with the average absolute prediction error of predictor r at time n, denoted $\bar{e}_r(n)$. Specifically, we define

$$w_r(n) = 2^{\max\{0, s_{\max} - c_n \bar{e}_r(n)\}} , \tag{6}$$

where $\bar{e}_r(n)$ is estimated using the exact same method and parameters of (4) on the sequence $e_r(i) = |\epsilon_r(i)|$, $i < n$, of past absolute errors from predictor r, $s_{\max}$ is a constant, and c_n is a pre-scaling factor that is doubled or halved at each time step if $W = \sum_r w_r(n)$ falls respectively below or above a range $[W_{\min}, W_{\max}]$; we fix $W_{\min} = 1$ and $W_{\max} = |\mathcal{P}|2^{(s_{\max}-1)}$ (this is, half the value that W would take if all the $|\mathcal{P}|$ predictors had their weights set to their maxima; see (6)). The above weighting scheme and its efficient implementation make up the second significant algorithmic contribution of this paper.

C. Additional performance improvements

A significant portion of the computational cost of the compression algorithm is spent on the update of its adaptive parameters. At the same time, we have observed that the adaptive parameters (Speck coefficients, predictor weights) tend to stabilize after a while, changing only when the statistical properties of the signal change significantly (e.g., at the beginning of a seizure). In order to avoid unnecessary updates, we track the performance of the overall scheme, and update the Speck coefficients and/or the predictor weights only when we observe a significant performance deterioration. In the case of the Speck coefficients, we update the coefficients of a predictor only when $e_r(n) > \bar{e}_r(n)$, with $e_r(n)$ and $\bar{e}_r(n)$ as defined following (6). In the case of the predictor weights, we update them every $T(n)$ samples. We begin with $T(0) = 1$. If none of the weights is effectively modified at time n, then $T(n+1) = \min\{2T(n), T_{\max}\}$. Otherwise, $T(n+1) = \max\{T(n)/4, 1\}$. The next update will be attempted at time $n + T(n+1)$.

In order to keep the computational complexity low, we selected a reduced ensemble of predictors that we have observed empirically to yield a good prediction performance. Specifically, the individual predictors (before weighting) used in the MCS algorithm are the following:

- a fixed 1st order predictor: $\hat{x}_i(n) = x_i(n-1)$,
- a 4th order single-channel Speck predictor,
- a 2nd order multi-channel Speck predictor,
- a 4th order multi-channel Speck predictor.

Although the performance of the first predictor is in general poor compared to that of the other three, it provides robustness and fast adaptation to sudden statistical changes in the signal.

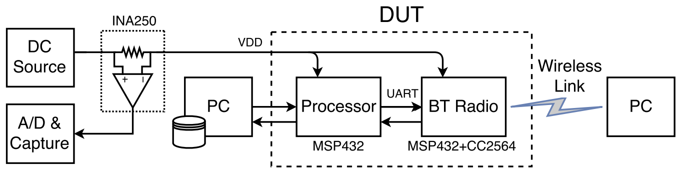

Fig. 4. Current consumption and throughput measurement setup.

D. MCF: Multi-Channel Fixed predictors

A significant speed up to the overall algorithm can be obtained at a cost of some compression degradation by replacing the adaptive Speck predictors with a set of simple fixed predictors that tend to work well on continuous signals:

- 1^{st} order: $\hat{x}_i(n) = x_i(n-1)$,
- 2^{nd} order: $\hat{x}_i(n) = 2x_i(n-1) - x_i(n-2)$,
- 3^{rd} order: $\hat{x}_i(n) = 3x_i(n-1) - 3x_i(n-2) + x_i(n-3)$,
- bilinear: $\hat{x}_i(n) = x_i(n-1) + x_\ell(n) - x_\ell(n-1)$.

We report on this algorithm, termed MCF, alongside MCS, in Section V.

E. Near-lossless encoding

In a near-lossless setting each prediction error, $\epsilon_i(n)$, is mapped before encoding to a quantized version, $\tilde{\epsilon}_i(n)$, defined as

$$\tilde{\epsilon}_i(n) = \text{sign}(\epsilon_i(n)) \left\lfloor \frac{|\epsilon_i(n)| + \delta}{2\delta + 1} \right\rfloor , \tag{7}$$

where $\lfloor z \rfloor$ denotes the largest integer not exceeding z. This quantization guarantees that the reconstructed value, $\tilde{x}_i(n) \triangleq \hat{x}_i(n) + \tilde{\epsilon}_i(n)(2\delta + 1)$, differs by up to δ from $x_i(n)$. All model parameters and predictions are calculated with $\tilde{x}_i(n)$ in lieu of $x_i(n)$, on both the encoder and the decoder side. Thus, the encoder and the decoder calculate exactly the same prediction for each sample, and the distortion originated by the quantization of prediction errors remains bounded in magnitude by δ (in particular, it does not accumulate over time).

IV. Modeling of power consumption

In this section we analyze and model the effect of different compression algorithms on the power consumption of the proposed platform. The power consumption of the AFE and ADC stages depends exclusively on the input data rate, i.e., the sampling frequency and number of channels. We thus focus on the power consumption of the processor block, which depends on the complexity of the compression algorithm, and on the power consumption of the BT radio block, which depends on the data rate output by the compressor. In the following subsections we propose simple models for the power consumption of each of these two blocks and we assess these models by measuring current consumption on actual hardware.

A. Measurement setup

The general setup used to measure the current consumption of both the processor and the BT radio is presented in Figure 4. A shunt resistor is placed in series with the dc power (3.3 V)

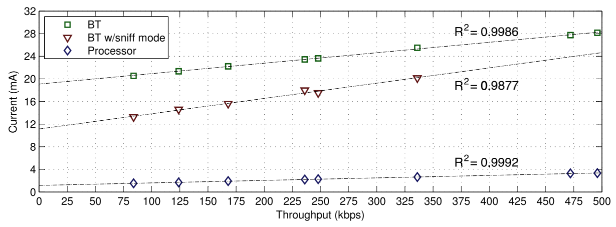

Fig. 5. Current consumption vs. data rate for processor (MCF, lossless compression) and BT models.

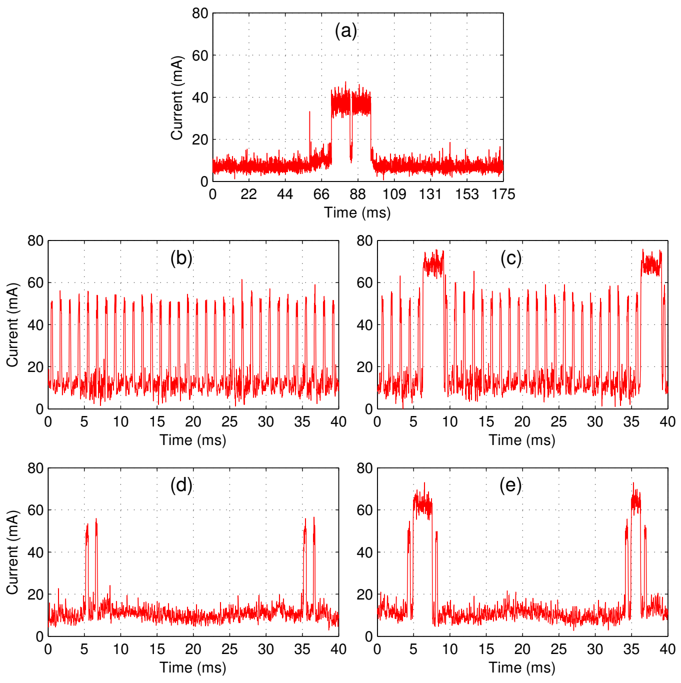

Fig. 6. Current consumption vs. time for different Bluetooth transmission states: (a) idle, (b) connected, (c) transmitting, (d) connected with sniff mode, and (e) transmitting with sniff mode. Comparing (b) with (d), and (c) with (e), an important reduction of current consumption peaks can be seen while the system operates with sniff mode.

to measure the current consumption. The voltage drop across the shunt is amplified and acquired with a 12-bits ADC board connected to a PC. The data is fed to the platform via a USB interface from a PC, and the data rate is controlled by the combination of an internal timer of the processor and hardware flow control, used to signal the PC when a new sample can be received. Upon completing the compression of a vector sample the processor enters in sleep mode until the timer expires.

The EEG data used to perform the experiments reported in this section and in Section V are taken from three different public databases, each obtained from a different subject, with a different acquisition hardware, and a different number of channels. The EEG signals, all originally sampled at 1 kHz, were downsampled to obtain 250 Hz and 500 Hz versions. We provide detailed information on these databases in Section V.

B. *Processor power consumption*

The current consumption of the microcontroller can be accurately estimated as the sum of the current consumption of active and sleep modes weighted by the respective duty-cycle. Therefore, since the execution times for both MCS and MCF are of linear order in the number of scalar samples, the current consumption of the processor block is presumably well approximated by a linear function of the input data rate.

To confirm this assumption, we executed the compression algorithms for different configurations of sampling rate and number of channels, and we measured the active and sleep time for the duty-cycle computation using the EnergyTrace+ tool included in the Code Composer Studio v6.1.2 (CCS) integrated development environment (IDE) from Texas Instruments. The current consumption during the active mode was measured while continuously compressing data, with the sleep mode disabled. The sleep mode current was measured by forcing the microcontroller into this state. The results were 4.27 mA for active mode and 1.13 mA for sleep mode.

The bottom dotted line in Figure 5 shows the estimated current consumption for the lossless MCF algorithm together with a linear fit as a function of the input data rate. As can be seen, the model fits the data very well. The remaining curves in Figure 5 are discussed in the sequel.

C. *BT radio power consumption*

The power consumption of BT depends on the state of the link, which can be any of *idle*, *connected*, or *transmitting*. Figure 6 shows samples of current consumption over time for each of these states; the curves are characteristic of BT communications [29].

BT links in *active* state (i.e., connected and transmitting states) require periodic exchanges of packets in order to keep the connection active and synchronized. These transmissions can be seen as the peaks in the plot of Figure 6b; transmissions with actual payload can be seen as pulses in Figure 6c.

In *sniff mode*, the BT device transmits/receives only at certain regular time intervals and during a specific period. This allows the radio to enter a low-power mode between transmissions, which results in an energy saving in exchange for a smaller maximum attainable throughput and slightly larger latency. The power consumption in sniff mode for a sleep time period of 30 ms is shown in figures 6d and 6e; we notice a reduction in the number of current consumption peaks with respect to figures 6b and 6c.

To evaluate the current consumption of the BT radio alone, we modify the setup of Figure 4 by feeding the processor directly from the voltage source and measure the current drain. The processor is set to send the input samples directly to the BT radio (no compression) and to control the sampling rate as explained before, so that the output data rate coincides with the input data rate.

Figure 5 shows the current consumption of the BT radio for different configurations of sampling rate and number of channels, as a function of the data throughput, for both BT with sniff mode off and on. The figure also shows, in dashed lines, the plots of a linear regression for each of the sniff modes; we observe an excellent fit in both cases.

The power models presented here can be extrapolated to

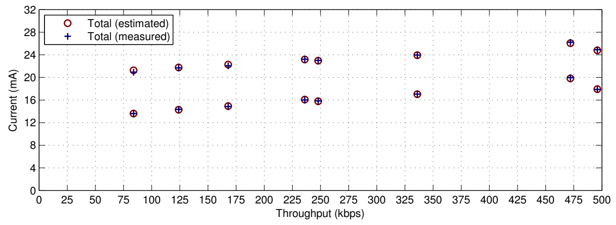

Fig. 7. Current consumption vs. input data rate for total measured and estimated current consumption (upper points correspond to BT sniff mode off, and lower points to BT with sniff mode on).

newer standards or technologies, such as BLE 5 or Wi-Fi. This can be done by plugging into the model the power consumption vs. throughput curve of the new wireless device or chipset.

D. Joint processor and BT radio power consumption

Figure 7 shows current consumption measurements of the processor and BT radio blocks for algorithm MCF and different configurations of number of channels and sampling rate, as a function of the input data rate, both for sniff mode off and on. The figure also shows an estimation of the current consumption calculated as the sum of separate estimations for the processor and BT radio blocks. For the former estimate, we determined the input data rate as a function of the sampling frequency and the number of channels for each evaluated EEG signal, and used the model described in subsection IV-B. For the latter, we determined the output data rate by compressing offline each evaluated EEG signal, and applied the model of subsection IV-C. The estimates match the actual measurements in all cases. We also notice that the current consumption does not follow a linear relation with the input data rate, mainly because some of the tested EEG signals are more compressible than others. Specifically, let R denote the *compression ratio* (CR) achieved by a compression algorithm A for an EEG signal, defined as the average number of bits of encoding per scalar sample, and let s denote the input data rate in bits per second. Then, our estimate $\hat{C}(A)$ of the current consumption for the processor and BT radio blocks takes the form

$$\hat{C}(A) = \alpha_c s + \alpha_r \frac{Rs}{n_{\text{bits}}} + \gamma_c + \gamma_r \,, \tag{8}$$

where n_{bits} is the sample resolution in bits, α_c and α_r are the linear coefficients of the models for the current consumption of the processor and BT radio blocks, respectively, and γ_c and γ_r are the independent terms in these linear models. For an alternative compression algorithm A' with compression ratio R', where $R' > R$ (worse), and model parameters α'_c, γ'_c, with $\alpha'_c < \alpha_c$ (less complex), Equation (8) determines a threshold on $R' - R$,

$$\mathcal{R} = \frac{n_{\text{bits}}}{\alpha_r} \left(\frac{\gamma_c - \gamma'_c}{s} + \alpha_c - \alpha'_c \right) \,, \tag{9}$$

such that algorithm A is more energy efficient than A', i.e., $C(A) < C(A')$, as long as $R' - R > \mathcal{R}$. As expected, $\mathcal{R}$ decreases with α_r. For high throughputs, i.e., large s, the prevalent term in (9) is driven by the proportion between $\alpha_c - \alpha'_c$ and α_r, where the first term depends on the hardware that executes the compressor and the difference in algorithm complexity, while the second term depends on the wireless communication technology.

V. Experimental performance evaluation

We conduct experiments to evaluate the compression performance of MCS and MCF, their execution time and storage requirements in our low power platform, and the power consumption and the data throughput attainable by the platform using each of the algorithms. Both algorithm were initially developed for a desktop computer and the source code later ported to our platform and compiled with the GNU v4.8.4 (Linaro) compiler. For comparison purposes, we also ported and tested on our platform a C implementation of the algorithm in [10], which we refer to as FLO4. In this implementation, the maximum predictor order (P in (3)) is set to 4, rather than 7 as in [10], to make the memory requirements fit the MSP432 RAM. This order reduction causes a small compression performance deterioration, which we report on in Section V-B, but it allows for testing FLO4 in our platform.

All the EEG signals used in the experiments reported here are taken from publicly available databases:

- DB1a and DB1b [30], [31]: 64-channel, 160 Hz, 12bps EEG of 109 subjects using the BCI2000 system. Recordings are divided in 2-minute motor imagery task (DB1a) and 1-minute calibration (DB1b).
- DB2a and DB2b [32] (BCI Competition III): 118-channel, 1k Hz, 16bps EEG of 6 subjects performing motor imagery tasks (DB2a). DB2b is a 100 Hz downsampled version of DB2a.
- DB3 [33] (BCI Competition IV): 59-channel, 1k Hz, 16bps EEG of 7 subjects performing motor imagery tasks.
- DB4 [34]: 31-channel, 1k Hz, 16bps EEG of 15 subjects performing image classification and recognition tasks.

The measurement setting for power consumption evaluation is that presented in Section IV. Specifically, we used 21, 31, and 59 channel EEG signals from databases DB2,[3] DB4, and DB3, respectively. EEG signals at 250 Hz and 500 Hz were obtained by downsampling the original data.

A. Compression time and memory usage

TABLE I
Platform performance depending on the compression algorithm version ($\delta = 0$).

Alg.	Number of channels	Proc. time per sample (ms)	Max. sampling rate (sps)	RAM usage (kB)
MCS	21	0.432	2313	11.7
MCS	31	0.593	1686	14.8
MCS	59	1.232	812	23.4
MCF	21	0.286	3496	8.6
MCF	31	0.418	2394	10.1
MCF	59	0.826	1211	14.4

[3]We picked the channels that comprise the international 10-20 system [35].

To measure the compression time, the processor was isolated from the rest of the system and the software was modified so that the input samples were read from FLASH memory and the compression output was written to RAM memory. The time measurements were performed with the *Count Event tool* (included in the CCS IDE), counting machine cycles between two breakpoints and then obtaining the elapsed time by dividing the cycle count by the clock frequency. The clock frequency of the MSP432 was set at 48 MHz in all cases.

The platform performance, in terms of processing time and memory usage, is detailed in Table I for MCS and MCF. The third column shows the measured average time required to process all channels and the fourth column indicates the computed maximum sampling rate (calculated assuming that the microcontroller is always in active mode).

Results indicate that MCF shows a speedup of 40–50% relative to MCS, and also a lower usage of RAM memory (showed in column five). On the other hand, the FLASH memory usage is nearly constant in all cases, 27.5 kB and 26.6 kB for MCS and MCF, respectively.

Table I reports on the lossless versions of the compression algorithms ($\delta = 0$). The near-lossless versions ($\delta > 0$) do not increase the memory usage, and they increase the processing time by less than 3%.

Both MCS and MCF execute much faster than FLO4; the average compression time per scalar sample (CTPS) of FLO4 is almost 10 times larger than that of MCS on our platform. On a desktop PC (Intel i7, single threaded, 3.4GHz), the CTPS, measured including file I/O transfer, is more than 6 times larger for FLO4 than for MCS. For implementations of [10] setting $P = 3$ and $P = 2$, referred to as FLO3 and FLO2, the compression performance is not clearly better than that of MCS (see Section V-B) and, still, the CTPS on the same PC is 4.9 and 3.6 times larger than for MCS, respectively. Compared to the reference implementation of ALS[4] configured for compression ratio optimization (command line parameter -7), MCS is 160 times faster than ALS executing on the same PC. The command line parameter -z3 of the ALS implementation results, in general, in a slight degradation of the compression performance with respect to the results reported in Table II (see Section V-B), but the execution time is greatly reduced; even in this case, MCS is still more than 13 times faster.

B. Compression performance

For each database, each data file was compressed separately and the overall compression ratio (CR), in bits per sample (bps), was calculated as L/N_s, where N_s is the sum of the number of scalar samples over all files of the database, and L is the sum of the number of bits over all compressed files of the database; *smaller* CRs are better.

Table II shows the CRs and average CTPSs of MCS and MCF compared to those of FLO4, FLO3, FLO2, those reported in [10], and those obtained for ALS [16] (ALS attains the best CRs in [16], [19], [20], [36] for the same databases); these results are summarized in Figure 8, which shows the total average CR vs. average CTPS.

[4]http://www.nue.tu-berlin.de/menue/forschung/projekte/beendete_projekte/mpeg-4_audio_lossless_coding_als

TABLE II
COMPRESSION RATIO IN BITS PER SAMPLE (SMALLER IS BETTER) OF MCS, MCF ALGORITHMS FOR DIFFERENT DATABASES ($\delta = 0$) AND AVERAGE CTPS (μS). COMPARISON WITH STATE OF THE ART.

Algorithm	DB1a	DB1b	DB2a	DB2b	DB3	DB4	CTPS
MCS	4.82	4.94	5.34	6.97	5.47	3.81	0.08
MCF	5.09	5.18	5.96	7.41	5.90	4.35	0.05
FLO4	4.74	4.82	5.30	6.98	5.46	3.64	0.51
FLO3	4.76	4.85	5.36	7.03	5.51	3.72	0.39
FLO2	4.82	4.91	5.74	7.06	5.67	4.23	0.29
[10]	4.70	4.79	5.21	6.93	5.42	3.58	0.92
[16]	5.37	5.45	5.69	7.69	5.99	3.73	1.07

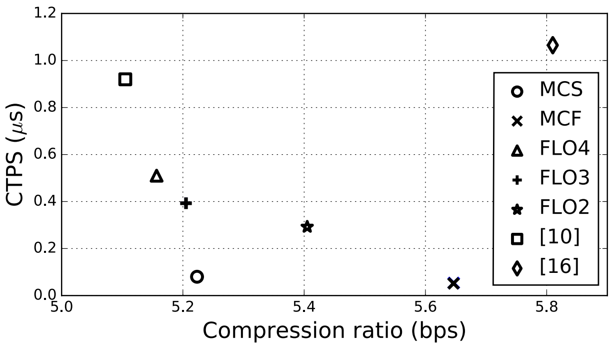

Fig. 8. Average compression ratio (bps) vs. complexity (CTPS).

MCS shows CRs that are very similar for some databases and higher (worse) than those of FLO4 in some cases. This deterioration is expected, due to the various simplifications made to lower the complexity of MCS, which, as detailed in Section V-A, results in a large efficiency gain. The compression performance of FLO3 is worse than that of MCS in half of the tested databases, and for FLO2 the compression performance is worse than that of MCS in almost all cases. As mentioned, however, the CTPS of MCS is still much lower than that of FLO3 and FLO2.

A compression performance deterioration is also observed in MCF with respect to MCS, due to the use of fixed predictors instead of adaptive ones, which, on the other hand, yielded important reductions in memory and time requirements as discussed in Subsection V-A. Notice, however, that the per-

TABLE III
COMPRESSION RATIO (BITS PER SAMPLE) OF MCS AND MCF ALGORITHMS FOR DIFFERENT DATABASES (SMALLER IS BETTER).

	δ	DB1a	DB1b	DB2a	DB2b	DB3	DB4
MCS	0	4.82	4.94	5.34	6.97	5.47	3.81
MCS	1	3.38	3.48	3.85	5.38	3.99	2.70
MCS	2	2.79	2.86	3.23	4.66	3.35	2.30
MCS	5	2.04	2.08	2.37	3.57	2.45	1.83
MCS	10	1.62	1.64	1.86	2.75	1.91	1.58
MCF	0	5.09	5.18	5.96	7.41	5.90	4.35
MCF	1	3.63	3.69	4.39	5.83	4.35	3.02
MCF	2	3.03	3.06	3.70	5.10	3.66	2.53
MCF	5	2.27	2.26	2.73	3.94	2.70	1.98
MCF	10	1.82	1.81	2.08	3.10	2.08	1.67

formance of MCF is still superior to that of the best algorithm reported in [16], [19], [20], [36] for all the databases except DB4. Comparing the CR of MCS with the original sample resolution for each database we observe that the amount of data that needs to be transmitted is reduced by a factor of at least 2.3 times, for DB2b, and up to 4.2 times, for DB4. The above conclusions are evident by inspecting Figure 8.

Finally, Table III shows near-lossless results for $\delta = \{1, 2, 5, 10\}$, including $\delta = 0$ from Table II as a reference, for both MCS and MCF.

C. Power consumption vs. throughput

Fig. 9 shows the current consumption of the platform (compression plus transmission) as a function of the data throughput for several values of the distortion parameter δ (shown next to the curve), different sampling rates (different color lines), and different number of channels (different markers). The curves on top correspond to BT with sniff mode off and the ones on the bottom to BT with sniff mode on. Fig. 9a shows the results for MCS and Fig. 9b for MCF. Current consumption was obtained by averaging the consumption during a time window between 20 and 80 seconds in the process of compressing and transmitting 20,000 samples; for each configuration of sampling rate and number of channels, the same EEG data file was used as input for both algorithms and for all values of δ. The dashed lines represent the current consumption of the BT radio alone transmitting raw (uncompressed) data.

We observe that the proposed low-power platform is able to perform lossless compression of a 59-channel acquisition at a rate of 500 sps, with a current consumption of 19.9 mA, that is 337 μA per channel (marked as 1 in Figure 9). Using near-lossless compression with distortion $\delta = 2$, in the same setting, results in almost 10% reduction in current consumption (marked as 2 in Figure 9). On the other hand, for 31 channels, a sampling rate of 1000 sps can be attained with a consumption of 590 μA/ch (marked as 3 in Figure 9).

Figure 9 illustrates the trade-off analyzed in Section IV-D between power invested in compression and power saved in data transmission. For very low data rates compressing does not pay off. For larger throughputs, however, the savings in transmission exceed the cost of compression, resulting in a reduction of up to 10% in current consumption. The figure also shows that the overall power consumption for MCF is in general smaller than for MCS, despite the compression performance of the latter being better. As explained in Section IV-D, this result depends on the specific hardware setting and the EEG compressibility.

The experiment also shows that, for a given current consumption, the proposed compression scheme results in a substantial increase in the maximum attainable throughput. For example, a budget of 24 mA allows for an uncompressed throughput of 265 ksps (see 4 in Figure 9), while the use of the MCF algorithm allows for throughputs of up to approximately 315 ksps for $\delta = 0$ (marked as 5 in Figure 9) and 400 ksps for $\delta = 2$ (marked as 6 in Figure 9). In other words, using data compression we obtain an increase in throughput of 19% with the lossless setting and 51% with the near-lossless one.

VI. Conclusions

We have presented a successful implementation, in a low-power wireless platform, of two lossless/near-lossless multichannel EEG compression algorithms that offer different levels of complexity and compression performance. We used these implementations to evaluate, experimentally, the energy saving and the increment in attainable throughput derived from the reduction in the amount of data transmitted; these turn out to be very significant for large throughput scenarios. Both algorithms are computationally efficient, yet they attain compression ratios that are very competitive with the best ones reported in the literature, which make them attractive also in other settings such as offline EEG compression.

Future work includes evaluating our low-power platform with other physiological signals such as electrocardiograms (ECG), for which the compression algorithm proposed here yields very promising results [10]. Another future objective is to develop a custom System on Chip using one of our algorithms to further reduce power consumption.

References

[1] G. Dufort, F. Favaro, F. Lecumberry, A. Martin, J. P. Oliver, J. Oreggioni, I. Ramirez, G. Seroussi, and L. Steinfeld, "Wearable EEG via lossless compression," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Aug 2016, pp. 1995–1998.

[2] V. Mihajlović, B. Grundlehner, R. Vullers, and J. Penders, "Wearable, wireless EEG solutions in daily life applications: What are we missing?" *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 1, pp. 6–21, Jan 2015.

[3] A. J. Casson, D. C. Yates, S. J. M. Smith, J. S. Duncan, and E. Rodriguez-Villegas, "Wearable electroencephalography," *IEEE Engineering in Medicine and Biology Magazine*, vol. 29, no. 3, pp. 44–56, May 2010.

[4] F. Zhang, J. Holleman, and B. Otis, "Design of ultra-low power biopotential amplifiers for biosignal acquisition applications," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 4, pp. 344–355, 2012.

[5] T. Y. Wang, M. R. Lai, C. M. Twigg, and S. Y. Peng, "A fully reconfigurable low-noise biopotential sensing amplifier with 1.96 noise efficiency factor," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 3, pp. 411–422, June 2014.

[6] A. M. R. Dixon, E. G. Allstot, D. Gangopadhyay, and D. J. Allstot, "Compressed sensing system considerations for ECG and EMG wireless biosensors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 2, pp. 156–166, April 2012.

[7] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst, "Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 9, pp. 2456–2466, Sept 2011.

[8] C. J. Deepu, C. H. Heng, and Y. Lian, "A hybrid data compression scheme for power reduction in wireless sensors for IoT," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 2, pp. 245–254, April 2017.

[9] S. A. Imtiaz, A. J. Casson, and E. Rodriguez-Villegas, "Compression in wearable sensor nodes: Impacts of node topology," *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 4, pp. 1080–1090, April 2014.

[10] I. Capurro, F. Lecumberry, Á. Martín, I. Ramírez, E. Rovira, and G. Seroussi, "Efficient sequential compression of multi-channel biomedical signals," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 4, pp. 904–916, July 2017.

[11] E. Spanò, S. D. Pascoli, and G. Iannaccone, "Low-power wearable ECG monitoring system for multiple-patient remote monitoring," *IEEE Sensors Journal*, vol. 16, no. 13, pp. 5452–5462, July 2016.

[12] R. Rieger and J. T. Taylor, "An adaptive sampling system for sensor nodes in body area networks," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 17, no. 2, pp. 183–189, April 2009.

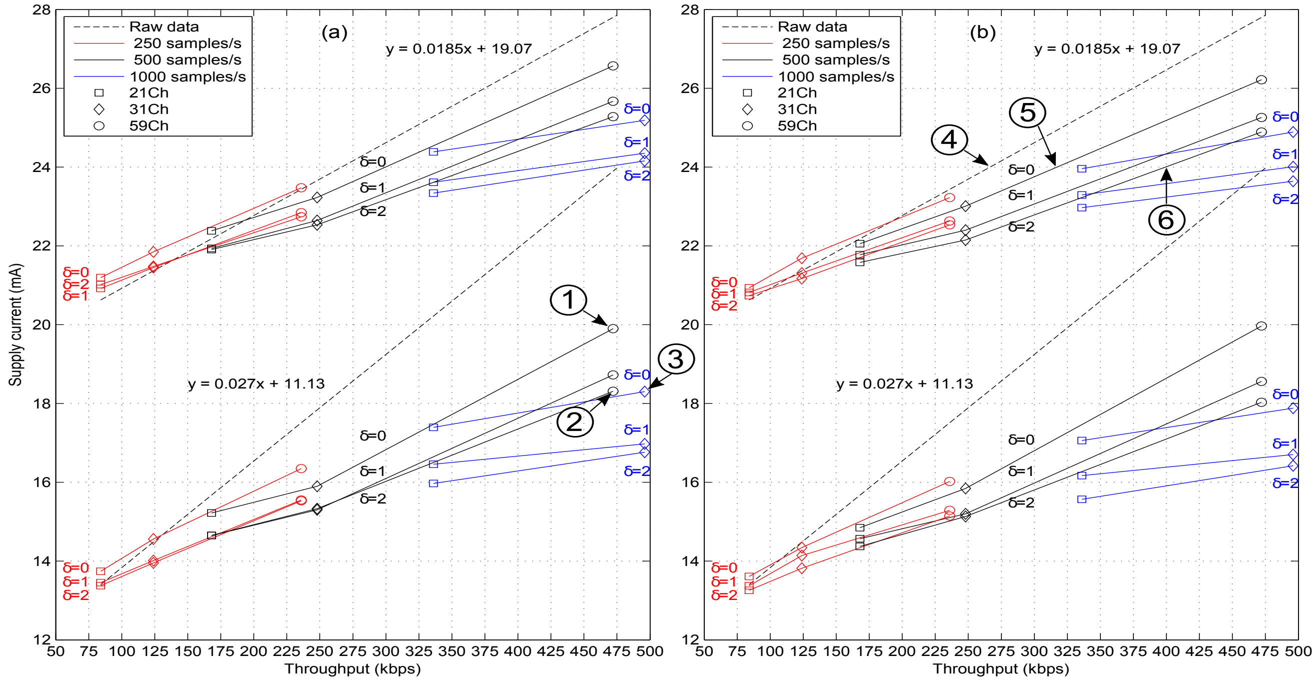

Fig. 9. Current consumption vs. throughput. (a) MCS, BT (top) and BT with sniff mode on (bottom). (b) MCF, BT (top) and BT with sniff mode on (bottom). The dashed line represents the current consumption as a function of the *uncompressed* data throughput.

[13] Y. Zhang, F. Zhang, Y. Shakhsheer, J. D. Silver, A. Klinefelter, M. Nagaraju, J. Boley, J. Pandey, A. Shrivastava, E. J. Carlson, A. Wood, B. H. Calhoun, and B. P. Otis, "A batteryless 19 muw mics/ism-band energy harvesting body sensor node soc for exg applications," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 1, pp. 199–213, Jan 2013.

[14] S. D. Pascoli, D. Puntin, A. Pinciaroli, E. Balaban, and M. Pompeiano, "Design and implementation of a wireless in-ovo EEG/EMG recorder," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, no. 6, pp. 832–840, Dec 2013.

[15] R. Harrison and C. Charles, "A low-power low-noise CMOS amplifier for neural recording applications," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 6, pp. 958–965, June 2003.

[16] ISO/IEC 14496-3:2005/Amd.2:2006, Information technology—Coding of audio-visual objects—Part 3: Audio, 3rd Ed. Amendment 2: Audio Lossless Coding (ALS), new audio profiles and BSAC extensions.

[17] Y. Kamamoto, N. Harada, and T. Moriya, "Interchannel dependency analysis of biomedical signals for efficient lossless compression by MPEG-4 ALS," in *Acoustics, Speech and Signal Processing, ICASSP 2008. IEEE International Conference on*, March 2008, pp. 569–572.

[18] Y. Wongsawat, S. Oraintara, T. Tanaka, and K. Rao, "Lossless multi-channel EEG compression," in *Proc. 2006 IEEE Int. Symp. Circuits and Systems*, May 2006.

[19] K. Srinivasan, J. Dauwels, and M. Reddy, "Multichannel EEG compression: Wavelet-based image and volumetric coding approach," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 1, pp. 113–120, Jan 2013.

[20] J. Dauwels, K. Srinivasan, M. Reddy, and A. Cichocki, "Near-lossless multichannel EEG compression based on matrix and tensor decompositions," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 3, pp. 708–714, May 2013.

[21] Z. Arnavut and H. Koak, "Lossless EEG signal compression," in *Proc. 5th Int. Conf. Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control*, Sept 2009.

[22] K. Srinivasan, J. Dauwels, and M. R. Reddy, "A two-dimensional approach for lossless EEG compression," *Biomedical Signal Processing and Control*, vol. 6, no. 4, pp. 387 – 394, 2011.

[23] J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM Journal of Research and Development*, vol. 20, no. 3, pp. 198–203, May 1976.

[24] S. W. Golomb, "Run-length encodings," *IEEE Trans. Inform. Theory*, vol. 12, pp. 399–401, Jul. 1966.

[25] B. Carpentieri, M. J. Weinberger, and G. Seroussi, "Lossless compression of continuous-tone images," *Proceedings of the IEEE*, vol. 88, no. 11, pp. 1797–1809, Nov 2000.

[26] A. Singer and M. Feder, "Universal linear prediction by model order weighting," *IEEE Trans. Sig. Processing*, vol. 47, no. 10, pp. 2685–2699, Oct 1999.

[27] G.-O. Glentis and N. Kalouptsidis, "Efficient order recursive algorithms for multichannel least squares filtering," *IEEE Trans. Sig. Processing*, vol. 40, no. 6, pp. 1354–1374, 1992.

[28] D. Speck, "Fast robust adaptation of predictor weights from min/max neighboring pixels for minimum conditional entropy," in *Signals, Systems and Computers, 1995. 1995 Conference Record of the Twenty-Ninth Asilomar Conference on*, vol. 1, Oct 1995, pp. 234–238 vol.1.

[29] D. Macii and D. Petri, "An effective power consumption measurement procedure for bluetooth wireless modules," *IEEE Transactions on Instrumentation and Measurement*, vol. 56, no. 4, pp. 1355–1364, Aug 2007.

[30] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, 2000 (June 13).

[31] G. Schalk, D. McFarland, T. Hinterberger, N. Birbaumer, and J. Wolpaw, "BCI2000: a general-purpose brain-computer interface (BCI) system," *IEEE Trans. Biomedical Engineering*, vol. 51, no. 6, pp. 1034–1043, June 2004.

[32] G. Dornhege, B. Blankertz, G. Curio, and K. Muller, "Boosting bit rates in noninvasive EEG single-trial classifications by feature combination and multiclass paradigms," *IEEE Trans. Biomedical Engineering*, vol. 51, no. 6, pp. 993–1002, June 2004.

[33] B. Blankertz, G. Dornhege, M. Krauledat, K.-R. Müller, and G. Curio, "The non-invasive Berlin brain–computer interface: Fast acquisition of effective performance in untrained subjects," *NeuroImage*, vol. 37, no. 2, pp. 539 – 550, 2007.

[34] A. Delorme, G. A. Rousselet, M. J.-M. Macé, and M. Fabre-Thorpe, "Interaction of top-down and bottom-up processing in the fast visual analysis of natural scenes," *Cognitive Brain Research*, vol. 19, no. 2, pp. 103 – 113, 2004.

[35] "Report of the committee on methods of clinical examination in electroencephalography," *Electroencephalography and Clinical Neurophysiology*, vol. 10, no. 2, pp. 370 – 375, 1958.

[36] B. Hejrati, A. Fathi, and F. Abdali-Mohammadi, "Efficient lossless multi-channel EEG compression based on channel clustering," *Biomed. Signal Process. Control*, vol. 31, pp. 295–300, 2017.

Guillermo Dufort y Álvarez was born in Montevideo, Uruguay. He received the computer engineer degree in informatics from Universidad de la República, Uruguay, in 2016. Since 2015, he has been with the Instituto de Computación, Universidad de la República. His research interests include information theory, machine learning, and bioinformatics.

Federico Favaro was born in Montevideo, Uruguay. He received the B.Sc degree in electrical engineering from Universidad de la República in 2016. He is currently a M.Sc student in electrical engineering and a teaching assistant with the Instituto de Ingeniería Eléctrica, Universidad de la República. His research interests include low power electronics, digital design, embedded systems, and biomedical applications.

Federico Lecumberry was born in Montevideo, Uruguay. He received the B.Sc., M.Sc. and Ph.D. degrees in Electrical Engineering from the Universidad de la República, Uruguay, in 2000, 2006 and 2012 respectively. He is an Associate Professor in Signal Processing with the Instituto de Ingeniería Eléctrica, Universidad de la República. He is also Principal Investigator of the Signal Processing Laboratory at the Institut Pasteur de Montevideo. His research interests include Signal Processing, Computer Vision, Machine Learning and Biomedical images.

Álvaro Martín was born in Montevideo, Uruguay. He received the computer engineer degree and the Ph.D. degree in informatics from Universidad de la República, Uruguay, in 2001 and 2009 respectively. Since 2000, he has been with the Instituto de Computación, Universidad de la República. His research interests include information theory, statistical modeling, and algorithms.

Juan P. Oliver received the B.Sc., M.Sc. and Ph.D. degrees in Electrical Engineering from the Universidad de la República, Uruguay, in 1989, 2007 and 2015 respectively. He is currently full time Associate Professor and Head of Electronics Department with the Instituto de Ingeniería Eléctrica, Universidad de la República, Uruguay. His research interests include the design of FPGA-based systems, low-power techniques, embedded systems, and electrical engineering education.

Julián Oreggioni received the B.Sc. and M.Sc. degrees in electrical engineering from Universidad de la República, in 2006 and 2013 respectively. He is currently Assistant Professor with the Instituto de Ingeniería Eléctrica, Universidad de la República (Uruguay). He has more than 10 years of experience in the electronics industry (embedded systems, M2M apps, vending machines, agrotech, etc.), he holds several patents and is coauthor of many technical articles. His research interests includes ultra low-power analog circuit and systems design and low-power embedded systems for biomedical and agricultural applications.

Ignacio Ramírez received the Electrical Engineer and the M.Sc. in Electrical Engineering degrees from the Universidad de la República in 2002 and 2007 respectively, and the Ph.D. degree in Scientific Computing from the University of Minnessota in 2012. He is with the Universidad de la República since 1999, where he now holds an Assistant Professor position in the Signal Processing Department. He is categorized as a Degree 1 Researcher by the National System of Researchers (SNI) and as a Degree 3 Professor in Mathematics by Programa de Desarrollo de las Ciencias Básicas (PEDECIBA). His research focuses in the development and application of Statistics, Information Theory and Optimization tools to signal/data processing and machine learning problems.

Gadiel Seroussi (M'87 - SM'91 - F'98) received the B.Sc. degree in electrical engineering and the M.Sc. and D.Sc. degrees in computer science from Technion-Israel Institute of Technology, Haifa, Israel, in 1977, 1979, and 1981, respectively. From 1981 to 1987, he was with the faculty of the Computer Science Department at Technion. From 1986 to 1988, he was a Senior Research Scientist at Cyclotomics Inc., Berkeley, CA. In 1988 he joined Hewlett-Packard Laboratories, Palo Alto, CA, where he founded the Information Theory Research Group and was its director until 2005. During the 2005—2006 academic year, he was an Associate Director of the Mathematical Sciences Research Institute in Berkeley, California. He returned to HP Labs in 2007, serving as a consultant to the information theory group until 2013. He is currently with Xperi Corp., Los Gatos, California. Since 2004, he has held a joint appointment in Computer Science and Electrical Engineering at Universidad de la República, Montevideo, Uruguay. He is a coauthor of the book Elliptic Curves in Cryptography (1999), and a coeditor of Advances in Elliptic Curve Cryptography (2005), both published by Cambridge University Press. His research interests include the mathematical foundations and practical applications of information theory, error correcting codes, data compression, audio and image processing, and cryptography.

Leonardo Steinfeld received the B.Sc., M.Sc. and Ph.D. degrees in Electrical Engineering from the Universidad de la República, Montevideo, Uruguay, in 2002, 2007 and 2013, respectively. He is currently an Assistant Professor with the Electronics Department at the Facultad de Ingeniería, Universidad de la República (Uruguay). His primary research interests includes low-power embedded systems and wireless sensor networks.