Vaquerizo-Hdez *et al. J Wireless Com Network* (2021) 2021:172
https://doi.org/10.1186/s13638-021-02043-w

EURASIP Journal on Wireless Communications and Networking

RESEARCH **Open Access**

Continuous energy consumption measure approach using a DMA double-buffering technique

Daniel Vaquerizo-Hdez, Pablo Muñoz, David F. Barrero and Maria D. R-Moreno*

*Correspondence: malola.rmoreno@uah.es
Universidad de Alcalá, Escuela Politécnica Superior, ISG, 28805 Alcalá de Henares, Madrid, Spain

Abstract
Measuring the consumption of electronic devices is a difficult and sensitive task. Data acquisition (DAQ) systems are often used to determine such consumption. In theory, measuring energy consumption is straight forward, just by acquiring current and voltage signals we can determine the consumption. However, a number of issues arise when a fine analysis is required. The main problem is that sampling frequencies have to be high enough to detect variations in the assessed signals over time. In that regard, some popular DAQ systems are based on RISC ARM processors for microcontrollers combined with analog-to-digital converters to meet high-frequency acquisition requirements. The efficient use of direct memory access (DMA) modules combined with pipelined processing in a microcontroller allows to improve the sample rate overcoming the processing time and the internal communication protocol limitations. This paper presents a novel approach for high-frequency energy measurement composed of a DMA rate improvement (data acquisition logic), a data processing logic and a low-cost hardware. The contribution of the paper is the combination of a double-buffered signal acquisition mechanism and an algorithm that computes the device's energy consumption using parallel data processing. The combination of these elements enables a high-frequency (continuous) energy consumption measurement of an electronic device, improving the accuracy and reducing the cost of existing systems. We have validated our approach by measuring the energy consumed by elemental circuits and wireless sensors networks (WSNs) motes. The results indicate that the energy measurement error is less than 5% and that the proposed method is suitable to measure WSN motes even during sleep cycles, enabling a better characterization of their consumption profile.

Keywords: ZigBee, WSN, Energy measurement, Double-buffering, DMA, DAQ

1 Introduction

Energy consumption is a critical aspect in the development of electronic circuits. One of the priority tasks when designing electronic devices is to know the consumption in order to select the components that supply the energy and to define the compatibility with other devices. The energy consumption is an important characteristic in data sheets and a source of information used to compare devices.

Measuring the energy consumption is not an easy task for various reasons. First, the energy consumption signals often do not have a defined frequency; they are quite random and often they vary very with high frequency, requiring costly measurement devices. Second, these signals contain noise that needs to be filtered. Third, the signals usually have low power, so errors are often introduced when using commercial devices to measure them, e.g., burden voltage.

Notwithstanding, the process for measuring these signals are not complicated when the right tools are used, such as digital signal processors (DSPs). However, these tools are not always available due to their big size and high cost. Therefore, other techniques need to be used.

When sizes are minimized and costs are reduced, we can measure the energy consumption incorporating small data acquisition (DAQ) systems instead of using DSPs. DAQ systems gather the energy consumption signal, operate the sampled data and they deliver the result to the user using data acquisition boards and programmable measuring instruments. Also, DAQ systems are commonly embedded in the consumer electronics that we use every day, providing us of their consumption in real time, e.g., mobile phones, tablets, smart watches, etc.

By getting deeper into the DAQ systems, we can see in the literature [10] that there are two processes required to know the value of a physical phenomenon:

1. The data acquisition process. This process obtains discrete samples periodically since the original signal is analog and it usually needs discrete data in a selected range.
2. The data handling process. This process uses the discrete data to obtain values or characteristics of the original signal.

Besides, depending on whether DAQ systems acquire data once or cyclically, they can be divided into the following types:

- "One-shot" DAQ system. They only analyze one set of samples on a time when the user requests it. These systems are useful if the sample to acquire is fixed, such as the samples from biochemical labs [39].
- "Continuous" DAQ system. They take some samples, analyzing them later. After that, a new set of samples is obtained and processed in an ongoing basis without requiring user interaction. These systems are useful if the signal to characterize varies over time such as acquiring biometeorological variables [6].

These two DAQ systems show different behaviors. The latency problems of the "one-shot" type have no influence in the measurement process. Instead, the latency problems of the "continuous" type influence the measures. The "continuous" type must pay particular attention to overlapping times in order to obtain real-time measures. This is because the "continuous" method analyzes the acquired samples before taking new samples. When the samples are being analyzed, the data acquisition process stops and the signal value is missed until the data handling process finishes. This creates uncertainty due to the latency of the data handling process and a discontinuity in the measured signal.

In order to solve the discontinuities in the measurement, as pointed out in the literature [4, 19], the data acquisition process can be deployed in parallel with the data handling process. Then, the data gathered is analyzed in sets meanwhile new data is gathered in parallel, i.e., the acquisition process is continuous. Using parallelism there is no latency due to the data processing of previous sample sets.

Using the advantages of the parallel deployment of the two processes, the approach presented in this paper measures the energy consumption of a device acquiring the current input signal of the device continuously. This approach was developed to solve the problems found during the testing of a WSN algorithm [37]. During the tests, we needed to measure the energy consumption of a ZigBee mote that uses an algorithm for reducing the energy consumption. To characterize the advantage of our algorithm we need to measure the energy consumption sampling it with a high temporal resolution (as the mote activation period is very short), and therefore we could not measure the energy consumption with standard laboratory tools. We would have required an expensive DSP and a lot of memory to solve the problem. Instead, we deployed the approach presented in this paper, which allowed us to measure and characterize the energy consumption of our motes.

Our approach uses the "continuous" measure method with a double-buffering technique to enable the parallel data processing of the signal. The novelty of the approach is the combination of a double-buffering DAQ system with a new data processing deployed in low-cost hardware. The results presented in this paper show how to obtain a high sample rate DAQ system using a pipeline processing composed of an ADC, two direct memory access (DMA) channels and a state machine with two timers, minimizing the data loss (maximizing the sample rate) when the data acquisition process discretizes the current input signal.

Meanwhile, the modern ADCs have high sample rates (up to giga samples per second), they are frequently used in microcontrollers with limitations. The nominal ADC sample rate is limited by the processing time and the communication time due to the memory access delay. In consequence, the effective ADC sample rate is less than the nominal ADC one, and some problems will almost certainly arise from this limitation such as data loss. In order to overcome this problem we can use two elements available in most microcontrollers that can improve the data transfer times and allow the parallel samples processing:

- The cache memory. It is a random access memory (RAM) faster than a regular RAM. In cache memory systems, the central processing unit (CPU) looks first in the cache memory; if it finds the data there, it does not have to do a more time-consuming reading from a slower memory or other data storage, since cache memory stores copies of the data frequently used. Using this memory, the microcontrollers can access more quickly to data that will be used several times. The microcontrollers use data caches to transfer data between slower and faster memories improving the access time.
- The direct memory access (DMA). It is an electronic device that allows accessing the memory independently of the CPU. With the DMA, the CPU is not interrupted when there is a memory transfer managed by the DMA. A DMA transfer

typically consists of copying a block of memory from one device to another, notifying the CPU when the transfer is completed.

The two previous elements allow us to improve memory access times and implement parallel processing. In the hardware we will use there is not internal cache memory, but it has a DMA unit. So, our approach uses the DMA to enable data transfer and the processing in parallel like Lewis [23] suggests. Attention should be drawn to the fact that our approach implements a double-buffer, allowing a predictable access time using the DMA, meanwhile the CPU can process the data without interruption. In this sense, the background of this paper is the development of high-performance routines for the data acquisition process and the data handling process running in parallel.

The contribution of this paper is an approach to calculate the energy consumption of a device supplied from a fixed DC voltage source, on a "continuous" way and with the lower data loss using a DAQ low-cost hardware. Specifically, our approach is composed of three parts: (1) a low-cost DAQ hardware, (2) a double-buffer data acquisition logic and (3) a data processing algorithm. In this regard, the paper presents a novel algorithm and two distributed mathematical calculation blocks. On the one hand, the algorithm implements the data acquisition process and the data processing in parallel. On the other hand, the mathematical calculation blocks minimize the operations of the data processing in order to reduce the processing time.

This approach allows using less powerful microcontrollers and/or reducing the required clock frequency when analyzing energy consumption and, therefore, reducing the cost. The results show that the error to measure energy consumption in our approach is less than 5% regarding the theoretical measure. Also, they probe that the on-chip ADC sampling rate limitations can be overcome, allowing the energy consumption measurement of WSN motes during sleep cycles.

This article is organized as follows. Section 2 includes the setup for the energy measurement. Next section introduces the state of the art of DAQ systems for energy consumption measuring. Then, Sect. 4 defines the problem of measuring energy for WSN motes. Section 5 provides an overview of the proposed energy measure approach. Following, Sect. 6 presents the hardware components used in our approach. Section 7 illustrates the process to acquire samples improving the sample rate through a double-buffered mechanism by means of the algorithm introduced in Sect. 7.2. Then, Sect. 8 shows how to analyze the samples in order to obtain the energy consumption. Section 9 states the limitations and temporal properties of the proposed approach. Section 10 presents the experiments performed to validate the proposed approach. Finally, a discussion, the conclusions and the future work are outlined.

2 Method

In our proposal for measuring the energy for the WSN motes, we need the following elements:

1. Hardware components. The hardware components are required to support signal sampling and data processing. They contain the necessary elements to convert the

input current into a voltage, to filter the signal, to discretize it, to store the samples, and to implement mathematical processes as well as data interchange protocols.

2. Data acquisition logic. The data acquisition logic is a software process that takes samples without stops. It uses DMA channels, timers and ADC routines to avoid interrupting the CPU, saving processing time and internal memory communication time.
3. Data processing. The data processing is a mathematical method that analyzes the latest samples taken by the data acquisition logic and calculates the energy consumption. It stops when the calculations are finished until new calculations are required. To perform the calculations it uses numerous equations in order to reduce computational cost, minimizing the microcontroller power and the processing time.

3 State of the art

In the state of the art we can appreciate different hardware solutions coupled with data acquisition and data handling processes for measuring the energy consumption of an electronic device. In this section we will review the hardware required to acquire signals and the mentioned processes.

3.1 Hardware solutions

Some hardware solutions are needed in DAQ systems to take samples periodically. The hardware used for this purpose can be divided into transducers, ADCs and microcontrollers, in conjunction with amplifiers and filters. Using these elements, in the literature we can find different hardware solutions for energy measurement:

- By charging and discharging capacitors. The capacitors are set between the power supply and the device that we want to measure. Then, the energy consumption can be obtained through the charge and discharge cycle of these capacitors as Andersen et al. [1] suggested. We do not follow this method because it is an empirical solution that depends on the current leakages of the capacitors.
- By a programmable ammeter. Another possibility is to periodically measure the current input signal to determine the energy consumption. However, a programmable ammeter is not enough for our purpose since the time between measures is too long and the burden voltage of common ammeters is very high, with the subsequent loss of accuracy.
- By sensors based on the Faraday's law. These sensors are the Rogowski coil [31] and the current transformer [26]. They provide electrical isolation, enabling the measurement of currents that circulate in the devices with high voltage. These sensors are not the subject of our approach since they work with medium or low frequencies due to the hysteresis cycle that is represented by their components.
- By means of magnetic field sensors. The following sensors can be seen as current sensors that use magnetic fields in the μA–mA range: the current sensors based on the Hall-effect [30], the current sensors based on the tunnel magnetoresistive effect [21], the current sensors based on galvanomagnetic technology [9], the current sensors based on anisotropic technology [3], the current sensors based on giant technology [2] and those based on the fluxgate principle [22]. We do not use these sensors

since we need more precision than those found in the market to embed them in a prototype circuit board (PCB).

- By recent current sensors based on tunnel magnetoresistive effects. García et al. [38] present a new sensor based on a tunnel magnetoresistive effect that improves on the older sensors based on magnetoresistive effects since they provide electrical isolation and have self-heating properties due to the power dissipation on it. However, currently they are not available in the market.
- By the switching cycles of a switching regulator. If switching regulators are used instead of linear regulators, there are nearly linear relationships between the switching frequency and the load current over a wide dynamic range. This relationship implies that a fixed amount of energy is delivered per cycle. Therefore, the energy consumption can be calculated counting the rising edges of the connected inductor voltage as Dutta et al. [13] appoint with the *iCount* energy meter design. However, in our approach we use a linear regulator instead of a switching regulator.
- By debugging and trace probes. We can measure the energy consumption of microcontrollers using tools such as *I-Jet* [17] for Arm Cortex microcontrollers. These devices provide power-debugging via their Joint Test Action Group and Serial Wire Debug (JTAG/SWD) connection. Also, they can debug external power signals adding other complementary devices such as *I-scope*. These devices are based on an energetic analysis of the microncontroller's code such as the EnergyTrace tool for the MSP430 microcontrollers [36]. Nevertheless, the resolution of these devices is small to measure WSNs (160 μA in the I-Jet device) and we want to improve their sample rate (200ksps in the I-Jet device [16]).
- By current sensors based on Ohm's law. This option measures the current consumption of a shunt resistor in conjunction to a low noise *chopper* amplifier. The voltage drop across the shunt resistor is used as a proportional measure of the current flow as illustrate Ziegler et al. [40]. Sometimes, the accuracy of the shunt resistor is poor and it does not work well, but using a resistance with great precision the results are optimal. We have chosen this solution because the error can be less than 0.1%, the thermal drift can be less than 25 ppm/K and the range can be μA–mA. Moreover, it allows the highest frequencies due to the Ohm's law with small size and low-cost components.

3.2 Data acquisition processes

The double-buffering technique is the main method to sample data for the real-time DAQ systems. It is not a new technique [20]; for instance, we can mention the old DAQ system of Gay et al. [14] that presents a multiprocessor double-buffering technique. However, we only use one processor in our approach.

There are some DAQ systems with double-buffering techniques running an ARM processor in parallel with a field-programmable gate array (FPGA) in order to take samples and process them as the work of Li et al. [24]. However we use the same device to take samples and process them. Other ARM DAQ systems with double-buffering techniques incorporate external ADC devices in order to take samples faster and save them in a specific hardware [34]. Nevertheless, our approach improves the sample rate without any external ADC device.

In addition, there are multiple applications used in other disciplines that help double-buffering techniques to take samples (or transfer them between memories) while the data are processed in the same processor as Tan et al. [35] suggest. This double-buffering technique has been used in image processing [41], paralleling processing [25] or ultrasound imaging methods [4]. Sancho et al. [33] exploit this compute-transfer overlap using double-buffering techniques that require minimal resources in the local memory available to the cores. However, our approach includes a new pipeline hardware state machine to process data from each parallel buffer while these buffers are filled cyclically.

What is more, Davide Rossi et al. [32] introduce an ultra-low-latency, highly energy-efficient DMA engine optimized for clustered shared memory many-core systems. They compare the results exchanging information using DMA and concluded that memory exchanges using small parallel buffers are much faster. However, our approach uses this DMA exchange technique by adding a pipeline processing composed of timers, interruptions and a state machine. This pipeline processing allows us to set the sample rate for complete control over the DAQ.

3.3 Data handling processes

Related to the data handling process, we have considered that the current input supply signals are not constant. Therefore, we must look at the classic definition of electrical energy, taking into account the power consumption at each discrete time interval. This same process is carried out by the newest electrical smart meters [8].

The smart meters [28] work with alternate current, meanwhile we work with continuous signals. Even so, both its signals and those we obtain are random depending on the energy consumption at every moment. The difference is that we optimize the mathematical calculation in order to minimize the operations carried out by the microcontroller, improving the processing time.

The difference between the presented works and our approach is that our work joins previous works to implement a method for energy measurement as efficiently as possible in a continuous basis, with only one processor and without external off-chip ADC components when the voltage supply does not change. Our approach uses transducers and a data acquisition logic accompanied by a data processing, giving a real solution to the measurement of energy when the current input signals change very quickly. Our work can perform other tasks in parallel while measuring energy, contributing to solve the problem of real-time energy measurement by minimizing resources.

4 The addressed problem

This section defines the addressed problem by our energy consumption measure approach. Also we determine the magnitude of the measured signals and we define the sampling rate required.

Our initial objective was a method to measure the energy consumption of ZigBee motes (a class of WSN protocol). We observed that the current input signal of the ZigBee motes varied very quickly in short periods of time. This is because the motes are in an idle state when no data is sent and in a very short active state when they are transmitting data. In the idle state the motes consume a negligible amount of energy, but in the active state they consume a lot of energy in small periods of time. Therefore, the energy consumption signal of the motes is mainly contributed by short energy peaks.

There are tools for estimating the energy consumption of a sensor mote analyzing the software and the energy consumption of each component of the mote such as AVAKIS [11]. Notwithstanding, we wanted to continuously measure the energy consumption of the motes physically deployed in a real-time application. One possible approach to do so was to assess only the energy peaks, ignoring the energy consumption when idle. However, we wanted to retrieve as much information as possible about the energy consumption, including the energy consumption of any state. That is, we want to take into account the energy consumption of the idle state too.

The most important consumption of a mote become the consumption of the microcontroller and the consumption of the radio transceiver. Therefore, we can estimate the total current drain as the sum of the microcontroller current drain plus the radio transceiver current drain. We obtained the current drain values for industrial and academics motes from previous works [37], summarized in Table 1. In this regard, the order of magnitude of the current drain for the sleep state and the active state is µA and mA respectively.

In order to sample the current drain signal we also need to consider its spectrum. In this regard, Casilari et al. [5] study the characterization of the energy consumption signal. The spectrum of the signal depends on the duty-cycling algorithm of the mote and the size of the sensed data. On the one hand, the duty-cycling algorithm controls the state change between the active state and the idle state, and therefore, the energy peaks frequency. On the other hand, the size of the sensed data determines the energy peaks width. Casilari et al. analyze the current drain signal of commercial ZigBee motes using an oscilloscope with 50 KSPS (kilosamples per second). They divide the analysis based on the ZigBee protocol communication sequence: start-up, scan, association, transmission and sleep state (idle state). As can be seen in their analysis, they show the current drain signal of the start-up, scan, association and transmission using 50 KSPS, but they still cannot analyze the sleep state over the time with that sample rate.

Focusing on the family of Berkeley motes (*WeC*, *René*, *René2*, *Dot*, *Mica*, *Mica2Dot*, *Mica2* and *Telos* [29]), we can observe that the wake-up time of these devices are very

Table 1 WSN platforms hardware characteristics (from manufactures data sheet) working at 3,3 V

Platforms	CPU			Radio			
	Model	Current drain		Model	Current drain		
		Sleep (µA)	Active (mA)		Sleep (µA)	Active (mA)	Using TX/RX (mA)
Mica2	ATmega128L	15	8	TI CC100	< 1	0.030–0.105	10–27
Telos	MSP430F1611	5.1	1.8	TI CC2420	20	0.426	11–20
TinyNode	MSP430F1611	5.1	1.8	XE1205	< 1	0.85–1.10	14–75
EyesIFX	MSP430F1611	5.1	1.8	TDA5250	< 1	1.8	9–12

low (6–1000 μS). Therefore, we have to maximize the sample rate if we want to detect when the mote wakes-up for a short period of time, for instance, when the mote wakes-up to not to be ejected from the network due to a downtime timeout.

Therefore, maximizing the signal sampling frequency is a priority to measure the energy consumption of the sleep state. In this regard, we estimate that doubling the sample rate over 95 KSPS is sufficient to measure the energy consumption of the sleep state. In this paper we present an energy consumption measure approach that covers these specifications.

5 The general approach

This section provides an overview of the developed approach and the parts in which it is divided.

As explained before, traditional approaches do not measure the energy consumption of the motes at an optimum sampling rate, generating loss of relevant information. Therefore, we want a system that improves the sampling rate for properly characterizing the energy consumption of electronic devices. For this end, we propose an approach for analyzing the energy consumption of electronic devices by means of maximizing the sample rate of commercial microcontrollers for adequate data gathering.

The most important problem we had to overcome was to improve the real sample rate of our DAQ system. The commercial microcontrollers with an ADC have a nominal sample rate that was never reached due to the limitations of the CPU architecture and due to the communication protocol of the microcontroller. The CPU processing time of the microcontroller and the time that the memories transfer information generate latency in the availability of data found in the ADC registers. The useful sample rate of the ADC was impaired, and therefore, the temporal resolution was not enough for our energy consumption measure application.

As a result, we designed a new approach where the real sample rate is improved. The following sections explain step by step how we did it, from a general approach to the details of implementation. Our system contributes to solve the problem of calculating the energy consumption when the input current signal is not band-limited and the voltage supply is constant. The idea is to periodically sample the electronic device input current and to calculate the energy consumption in parallel when the voltage supply is constant, i.e., when an electronic device is supplied by a voltage regulator.

To cope with the problem, we decided to create a new and fast energy measurement approach. The new approach would meet the following requirements in addition to those set out above:

- To use low-cost hardware. A current challenge in consumer electronics is to reduce costs. Consequently, the technique should be implemented in low-cost hardware.
- Low-power consumption. In order to guarantee high autonomy of the batteries we need low energy consumption.
- Minimize the data loss and maximize the sample rate. The sample rate should be as high as possible, finding a compromise solution.

- Continuous measurement over time. This approach measures the energy consumption without storing large amounts of data and without delays in signal sampling over time.

In order to meet all the requirements, the approach consists of the three elements described in Sect. 2.

The two software processes run in a cyclic schema and are performed in parallel. To deploy the two processes we have designed an algorithm which is in charge of programming the DAQ system taking into account the hardware components and their characteristics. It forms the core of the approach and implements the sample rate improvement.

The following three sections delve into the hardware components and the two processes by describing the algorithm that implements the double-buffered mechanism in Sect. 7 and including the equations that computes the energy consumption in Sect. 8.

6 Hardware components

This section presents the steps required by the hardware to take samples of the target input signal. Each step is in charge of a hardware component, and therefore, each hardware component will be explained when its associated step is explained.

The following steps enable discrete sample acquisition of the input current to feed the data processing of Sect. 8:

1. Turn the current input signal to a voltage signal.
2. Filter the noise by capacitors.
3. Amplify the voltage signal.
4. Convert the analog signal into a digital signal.
5. Keep the discrete values in a memory.

For the first step, we need to work in voltage values. For this purpose, a shunt resistor is used to convert the current input signal in a directly proportional voltage through the Ohm's law. The resistor has low error (ppm) since it does not pervert much the original signal. In the second step, the noise due to the high frequency is filtered by capacitors since the noise distorts the value of the current measure. In the third step, the signal is amplified by a *chopper* amplifier to eliminate the offset voltage and the typical $1/f$ noise of the operational amplifier. In the fourth step, the analog signal is converted to a digital signal using an ADC. Finally, the digital values are periodically saved in a memory.

To address the above steps, we have used the commercial µCurrent device as Geng et al. [15] suggest. µCurrent is an electronic device that overcomes burden voltage problems of the commercial ammeters to allow current measures. Technically, the µCurrent device already incorporates all electronic devices to support the steps 1, 2 and 3. Indeed, it incorporates three shunt resistors for allowing different ranges of conversion depending on the gain of the Ohm's law. In this regard, the accuracy is less than 0.2% on µA and nA ranges, and less than 0.5% on mA ranges. Also, the µCurrent device incorporates a MAX4239 *chopper* amplifier and filter capacitors. We have decided to use µCurrent to expedite the deployment of the data acquisition logic and the data processing. The most important µCurrent characteristics are shown hereunder:

- Current ranges: 0–1250 mA/μA/nA. The ranges change depending on the selected shunt resistor.
- Burden voltage: 20 μV/mA for mA range, 10 μV/μA for μA range and 10 μV/nA for nA range.
- Output: 1 mV/mA for mA range, 1 mV/μA for μA range, 1 mV/nA for nA range.
- Resolution: 100 pA.
- Noise less than − 90 dBV.
- Total Harmonic Distortion (THD) less than − 60 dB.

Furthermore, we have chosen a Cortex M0+ microcontroller (LPC824) to support steps 4 and 5 adding the on-chip ADC and the on-chip memory of the LPC824. The LPC824 is a 32-bit ARM Cortex M0+ core microcontroller of NXP Semiconductors. We have chosen it since it has a 12-bit ADC with offset configuration and flexible full scale by the pinout, the ADC conversion rate is up to 1.2 MSamples/seg (nominal) and it has an on-chip ROOM API to control the ADC.

To store the sampled data, the LPC824 has 8KB Static Random Access Memory (SRAM) memory and 32 KB flash memory to record the program instructions. Also, it has a DMA with multiples channels and triggers to implement the algorithm, as well as timers to use the triggers. Nowadays, Cortex M0+ is the most energy-efficient processor available: 9.4 μW/MHz.

Figure 1 illustrates the hardware connection to cope with the previous steps. With the objective to turn the current input signal to a voltage signal (amplified and filtered), the μCurrent device is wired in series with the device that we want to assess. With the aim of obtaining the digital signal and keeping the samples in memory, the LPC824 is connected to the μCurrent output. Finally, the LPC824 is connected to a PC through a serial interface with the purpose of logging the energy consumption along the time. The μCurrent output is ground referenced and therefore the LPC824 and the μCurrent are connected at the same common electronic ground.

Attention should be drawn to the fact that the μCurrent selects the shunt-resistor to fix the current range of the signal. Also, the analog pins V_{refp} and V_{refn} set the full scale and V_{dd} sets the offset in order to achieve the highest resolution of the ADC depending on the voltage signal amplitude on the μCurrent output. Regarding the PC connection, a serial interface is used wiring the RxD and TxD pins of the microcontroller to the USB port of the PC. In this direction, the hardware is configured depending on the range of the current input signal and its temporal resolution. This configuration is explained in Sects. 7 and 8.

7 Data acquisition logic

The aim of this section is to describe the logic used for continuous sampling, allowing parallel data processing as presented in Sect. 8.

The data acquisition logic is a pipeline process consisting of a string of peripherals linked to the LPC824 that are programmed individually. In this sense, this section is divided into two subsections: Sects. 7.1 and 7.2. The first one deals with the LPC824's peripherals required for implementing the data acquisition logic, clarifying its characteristics and justifying its usage. The second subsection describes in detail the algorithm

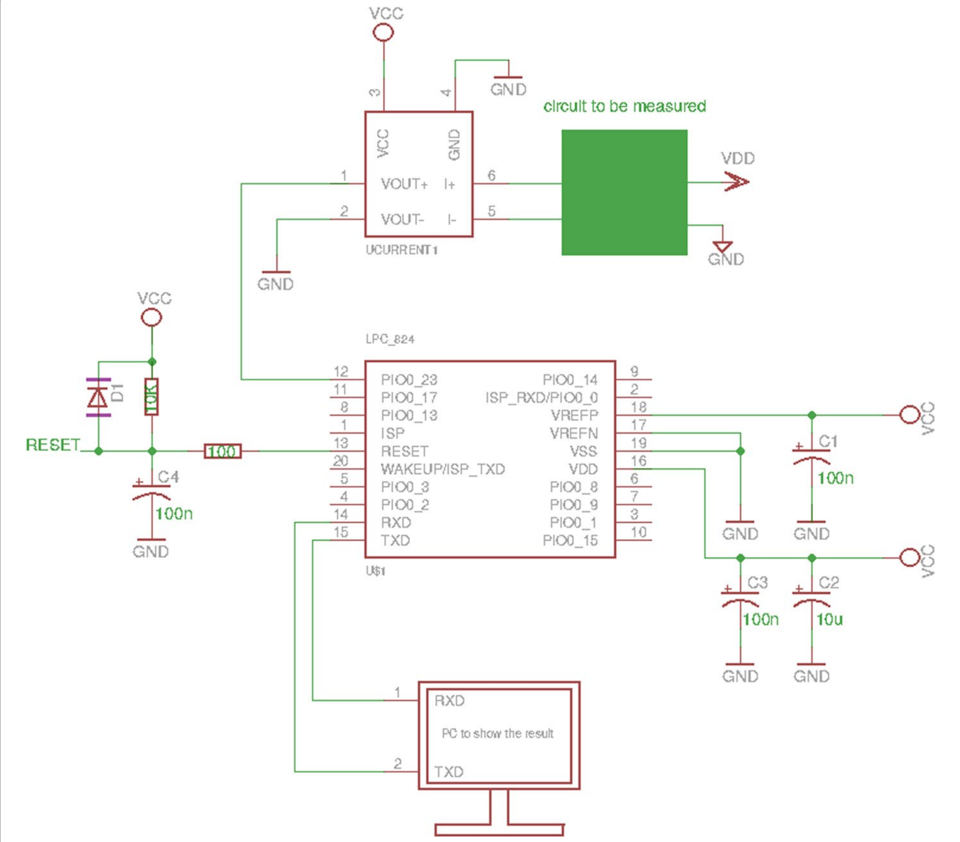

Fig. 1 Hardware components of the proposed approach for energy measurement

used to program the peripherals. To this end, the subsection contains a graph of the pipeline processing, a graph of the state machine deployed by the peripherals and a pseudocode of the algorithm.

It is worth mentioning that the data processing is executed by the algorithm. Thus, in this section we only indicate the places where the data processing is executed to make the exposition about the algorithm short. Subsequently, the data processing is described in Sect. 8.

7.1 Required peripherals

The peripherals characteristics of the data acquisition logic are shown below. First, they are named and then they are explained.

- A state-configurable timer (SCTimer).
- An ADC.
- Two DMA channels.
- Two buffers of SRAM.
- A universal asynchronous receiver–transmitter (UART).

In terms of synchronization, it is essential to know the sampled data rate. The mathematical method presented in Sect. 8 needs to be fed with the sample rate value. This value is fixed by the data acquisition logic, and this will assure the correct discretization of the original current input signal. To address it, the start point of the data acquisition logic is the LPC824's SCTimer as can be seen in Fig. 2. It is a peripheral designed to implement state machines that we use to set the sample rate of the ADC. Its configuration includes descending and ascending timers of 16 or 32 bits, events and states. The events are used to program the transition between states depending on if they are activated or not. These events can be activated by an external peripheral trigger or by the internal timers. Furthermore, the events can trigger external peripherals when they are activated.

When sampling the signal, the use of an ADC is essential to discretize the input signal since the technique operates digitally. Also, it saves the discrete value of the signal in a register for further processing. As can be seen in Fig. 2, the ADC samples data synchronized with the SCTimer.

Beside the ADC, the LPC824 has a DMA device with 18 channels that can be programmed independently. Each LPC824's DMA channel enables a memory data transfer. Each one can be programmed using their respective control registers to exchange data between memories and the LPC824's peripherals, such as the ADC. Regarding the initialization of the data transfer, 9 sources can trigger the DMA divided into internal events, pin interrupts, peripheral requests and DMA outputs. Our approach only uses two channels: channel_0 and channel_1 which are triggered by the ADC when it finishes to sample a datum.

Furthermore, the LPC824 includes an 8 kB on-chip static SRAM data memory in two separate blocks of 4 kB. Our approach uses the two 4 kB blocks in order to save the sampled data, deploying two buffers to manage data. Finally, the LPC824 has three UARTs to implement protocols to exchange information. In this sense, the technique uses a UART to send the measured information to a PC. The PC displays the measured energy consumption.

7.2 The algorithm to program the DAQ system

In this subsection we present the algorithm used to program the data acquisition logic using the peripherals through a pipeline process. Also, we have indicated in the algorithm that the data processing (Sect. 8) takes place on lines 39 and 42 of Algorithm 1.

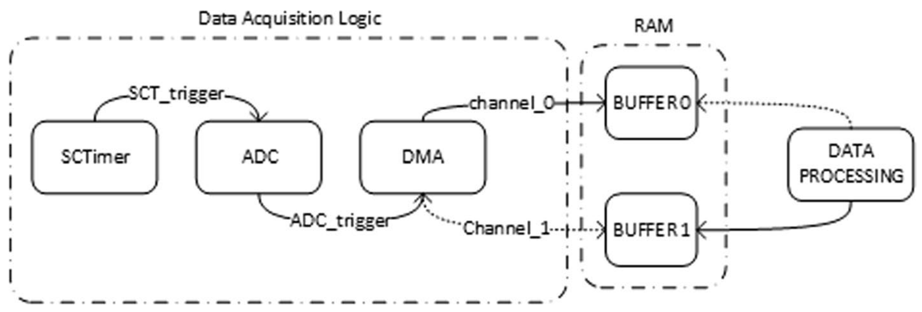

Fig. 2 Pipeline processing data acquisition logic and data processing implemented for a double buffer in the SRAM

Algorithm 1 Algorithm

```
function $CONFIGURE\_DMA()$   ▷ Configuration of the DMA device
    $DMA\_channel\_0\_\&\_1\_source \leftarrow ADC$   ▷ ADC origin for all transfers
    $DMA\_channel\_0\_destination \leftarrow BUFFER0$   ▷ Destination of the DMA channel_0
    $DMA\_channel\_1\_destination \leftarrow BUFFER1$   ▷ Destination of the DMA channel_1
    $DMA\ activation\ input \leftarrow ADC\_trigger$   ▷ 1 memory exchange = 1 ADC sample
    $enable\_channel\_0 \leftarrow TRUE$   ▷ channel_0 does 1st info exchange
    return
function $CONFIGURE\_ADC()$   ▷ Configuration of the ADC device. ADC can work as fast as possible & SCTimer selects ADC sample rate
    $maximum\_clock\_rate\_\&\_minimum\_clock\_divider \leftarrow TRUE$
    $ADC\ activation\ input \leftarrow SCT\_trigger$   ▷ 1 ADC sample = 1 SCT event_0
    $ADC\_trigger \leftarrow TRUE\ when\ an\ ADC\ sample\ finishes$
    return
function $CONFIGURE\_SCT()$   ▷ Configuration of the SCT device. SCT state machine has 2 timers, 1 state and 2 events
    $decrem\_count\_timer\_0 \leftarrow FREQUENCY$ ▷ Frequency = Sample rate
    $decrem\_count\_timer\_1\_at\_a\_freq \leftarrow 2 * FREQUENCYTRUE$
    $autoreload\_timer\_0\_\&\_1 \leftarrow TRUE$   ▷ 2 timers reset when count ends
    $state\_1 \leftarrow event\_0\_\&\_1$   ▷ State machine has 1 state composed of 2 events   ▷ Each event is associated with the a timer counter. There is an event if the count is 0
    $event\_0 = !timer\_0\ event \leftarrow priority\_HIGH$   ▷ !timer_0 event priority>!timer_1 event priority
    $event\_1 = !timer\_1\ event \leftarrow priority\_LOW$
    $event\_0\_\&\_1 \leftarrow match\_mode$   ▷ Events occur when the timers finish
    $SCT\_trigger\_set\ action \leftarrow event\_0$   ▷ Event 0 sets SCTimer trigger
    $SCT\_trigger\_clear\ action \leftarrow event\_1$   ▷ Event 1 cleans SCTimer trigger
    return
function $handle\ IRQ\_DMA()$   ▷ IRQ when a SRAM block transfer finishes
                ▷ Gets DMA_channel associated with IRQ & clean flags
    $Enable\_DMA\_channel = READ\_DMA\_CHANNEL\_FINISH\_TRANSFER\_FLAG()$
    $CLEAN\_DMA\_CHANNEL\_FINISH\_TRANSFER\_FLAG()$
    return
procedure MAIN   ▷ Beginning of the algorithm
    $CONFIGURE\_DMA(DMA\_channel\_0)$   ▷ Conf. all devices & IRQs
    $CONFIGURE\_ADC()$
    $CONFIGURE\_SCT()$
    loop
        while n_buffers > 0 do   ▷ n_buffers=number of buffers filled until send data to UART
            $WFI()$   ▷ Wait For Interrupt
            $n\_buffers--$
            if Enable_DMA_channel == DMA_channel_0 then
                $enable\_channel\_1 \leftarrow TRUE$ ▷ Channel_1 starts info transfer
                $DATA\_PROCESSING\_BUFFER0$   ▷ Data Processing of BUFFER0
            else Enable_DMA_channel == DMA_channel_1
                $enable\_channel\_0 \leftarrow TRUE$   ▷ Channel_0 starts the information transfer
                $DATA\_PROCESSING\_BUFFER1$   ▷ Data Processing of BUFFER1
        $Send\_UART(result)$▷ Send energy measure to PC through UART
        $n\_buffers \leftarrow RESET$   ▷ Reset n_buffers
```

The algorithm uses two DMA channels providing two data transfers channels between the ADC data register and two buffers in the SRAM memory. Each DMA channel has an associated SRAM buffer: *BUFFER0* and *BUFFER1*. As can been seen in Fig. 2 the idea is to fill the SRAM buffers with the values of the input current signal sampled. When one of them is being filled, the other is used to calculate the energy consumption and vice versa. These operations increase the real ADC sample rate while a continuous double buffer is allowed.

As stated previously, the algorithm uses the SCTimer with the objective of fixing the sample rate. The SCTimer is used to create a state machine that controls the memory transfer as can be seen in Fig. 3. In this direction, the SCTimer enables a state machine that triggers the ADC. The ADC obtains one sample when it is triggered by the SCTimer.

The state machine is composed of the following elements to trigger the ADC at a determined frequency:

- One state: *State1*.
- Two descending timers: *timer_0* and *timer_1*.
- Two events associated with each timer: *!timer_0* and *!timer_1*.
- An action to trigger the ADC: *SCT_trigger_set*.
- An action to clean the ADC activation input: *SCT_trigger_clean*.
- A trigger signal: *SCT_trigger*.

As illustrated in Fig. 3, the SCTimer is a Mealy state machine composed of one state, two inputs and two outputs. The inputs are the values of two autoreload descending timers (*timer_0* and *timer_1*). The outputs are two events associated with each timer (*!timer_0* and *!timer_1*). Each event is activated when its associated timer value is 0. The frequency of the *timer_1* is double that of the *timer_0*. And the *!timer_0* event priority is higher that the *!timer_1* event priority.

In order to trigger the ADC, the state machine uses a signal of the SCTimer peripheral (*SCTimer_trigger*), associating it with the ADC activation input by using the ADC configuration register. The ADC will take a sample when it finds a rising edge on its activation input synchronized with a rising edge in the *SCTimer_trigger*.

Therefore, if the *!timer_0* event is activated, the *SCT_trigger_set* action is activated setting the *SCTimer_trigger* to a high Boolean level and the ADC is activated. If the *!timer_1* event is activated, the *SCT_trigger_clean* action is activated switching the *SCTimer_trigger* to a low Boolean level and the ADC is deactivated.

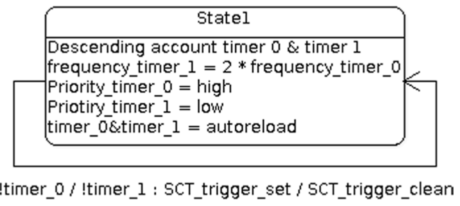
State1
Descending account timer 0 & timer 1
frequency_timer_1 = 2 * frequency_timer_0
Priority_timer_0 = high
Priotiry_timer_1 = low
timer_0&timer_1 = autoreload
!timer_0 / !timer_1 : SCT_trigger_set / SCT_trigger_clean

Fig. 3 SCTimer state machine. It has only one state with two autoreload downward timers and two events associated with each timer. The timer 1 is twice as fast as the timer 0. The priority of the !timer 0 event is higher than the priority of the !timer 1 event. Each event is activated when its associated timer value is 0. If the !timer 0 event is activated, the SCT trigger set action is activated. If the !timer 1 event is activated, the SCT trigger clean action is activated

The state machine activates the *SCTimer_trigger* at a given frequency. Then, the *SCTimer_trigger* will activate the ADC sampling at that frequency. In addition, on the one hand we make sure that the *SCTimer_trigger* is always activated per cycle by giving priority to the *SCT_trigger_set* action over the *SCT_trigger_clean* action. On the other hand, we ensure that there is a falling flank in the *SCTimer_trigger* between each sample of the ADC by allowing the frequency of the *SCT_trigger_clean* action to the double of the *SCT_trigger_set* action setting the *timer_1* frequency value to the double of *timer_0* value.

Concerning to the ADC, it is programmed to the maximum resolution and the highest sampling nominal rate, and to obtain one sample when it is triggered by the SCTimer. It has a trigger signal (*ADC_trigger*) to activate other peripherals when it finishes taking a sample. In this sense, the *ADC_trigger* is connected to the DMA activation input. When the ADC ends taking one sample, it generates a trigger in the two DMA channels to exchange information between the ADC register (where the sampled datum is saved) and the associated SRAM buffers.

Regarding the DMA, the two DMA channels are configured in a "one-shot" basis. This means that each DMA channel exchanges information once per request. The request occurs when the ADC finishes to take a sample and triggers the DMA. One DMA channel is always enabled and one DMA channel is always disabled. Therefore, the trigger produced by the ADC is listened only by the enabled DMA channel. When the DMA is triggered, one DMA channel fills its associated SRAM buffer with the data saved in the ADC register. When the associated SRAM buffer is completely filled, the DMA channel stops until the channel reset, while the other DMA channel is activated since its reset.

Each DMA channel fills up its associated SRAM buffer in a cyclic form alternatively, that is, first the DMA *channel_0* associated buffer (*BUFFER0*) and then the DMA *channel_1* (*BUFFER1*). The DMA channels are working alternately when they fill in their associated buffers; only one DMA channel will be exchanging data until its associated buffer is completely full while the other DMA channel is disabled. When a DMA channel fills out its associated buffer, an interruption generates a program exception in the CPU. Then, an Interrupt ReQuest (IRQ) handler gets what channel originated it and it cleans the associated flag. After that, the channel that did not cause the interruption is reset to be filled again while the other one is disabled. Furthermore, the data of the buffer in idle state are used by the data processing method (detailed in Sect. 8) to calculate the energy consumption.

It is worth mentioning that if we want to convert the approach to a "one-shot" DAQ system, we only have to eliminate the double-buffered part of the data acquisition logic. We can convert the approach to a "one-shot" system using only one DMA channel and its associated SRAM buffer, while the SCTimer only has one downward timer without autoreload of the count. This allows measuring a discrete specific temporary window of the energy consumption, which it is not the goal of the paper.

The Algorithm 1 describes the program flow embedded in the LPC824 microcontroller. It is composed of the main procedure in line 29, one IRQ handle in line 24 that is executed when the DMA finishes exchanging data to the DATA_PROCESSING_BUFFER0 (one of the SRAM buffers) or the DATA_PROCESSING_BUFFER1 (the other SRAM buffer), and three configuration functions that configure the peripherals:

the CONFIGURE_DMA() function in line 1, the CONFIGURE_ADC() function in line 8 and the CONFIGURE_SCT() in line 13. The CONFIGURE_DMA() function configures the DMA, the CONFIGURE_ADC() function configures the ADC and the SCT function configures the SCTtimer by the internal registers of the microcontroller.

The main procedure calls each configuration function at the beginning and then, the program stops in line 35 waiting for the IRQ DMA handle of the line 24 to be executed. Cyclically, there is one SRAM filled buffer and there is other SRAM buffer filling up: the *DATA_PROCESSING_BUFFER0* or the *DATA_PROCESSING_BUFFER1*. When the SRAM buffer *DATA_PROCESSING_BUFFER0* or the SRAM buffer *DATA_PROCESSING_BUFFER1* is full, the IRQ DMA handle is executed. After the IRQ DMA handle execution, the algorithm performs a data processing to the SRAM buffer that has just been filled to know the power consumption and enables the previously filled buffer to be refilled again. This procedure is performed in a cyclic manner until a filled buffer number is completed. The maximum number of completed buffers is indicated with the *n_buffers* variable in line 34. The following is a detailed explanation of the entire program thread.

The Algorithm 1 starts from line 29 by configuring the DMA, the ADC and the SCTimer (functions on lines 1, 8 and 13). The *CONFIGURE_DMA()* configures the information transfer, selecting the source (line 2), the destination (lines 3 and 4), it matches the DMA activation input with the *ADC_trigger* (line 5) and finally it enables one of the two channels for the first information exchange (line 6). The *CONFIGURE_ADC()* function selects the highest ADC speed configuration to take samples (line 9) and it matches its ADC activation input with the *SCTimer_trigger* signal (line 10). The *CONFIGURE_SCT()* function configures the ADC sample rate (lines 14 and 15) synchronizing the *timer_0* frequency and the *timer_1* frequency taking into account that the *timer_1* frequency value is the double of the *timer_0* frequency value. Also, it sets the autoreload mode for each timer (line 16), configures the priority of the events (lines 18 and 19) and matches the events with the actions to trigger the ADC (lines 21 and 22). That is, this function implements the Mealy state machine.

Once the DMA, the ADC and the SCTimer are configured, a buffer is filled while the program thread is stopped in the *WFI()* function (line 35). When the buffer is full, it generates a program exception with an interruption captured by the handler defined in line 24. This handler acknowledges which DMA channel generated the interruption and therefore, which buffer is full. Then, the program continues on lines 38 or 41 to reset the channel that did not generate the last program exception, while the other channel is disabled.

Then, the data processing of the buffer that generated the last exception begins in parallel to the data acquisition. The program processes the last filled buffer (line 39 or 42). While a buffer is processed, the other buffer is being filled until the DMA interruption. It is worth mentioning that there is only one thread in the program.

The variable *n_buffers* (line 34) is the maximum number of filled buffers that the data processing allows before sending the measures to the PC via the UART. This number depends on the SRAM buffer size and the ADC sample rate. *n_buffers* will be calculated in Sect. 9. Then, *n_buffers* is reset and the measure of the energy consumption continues.

8 Data processing

In the previous section, the data acquisition logic takes samples that must be processed in order to compute the energy consumed. In this regard, this section explains the method to allow the data processing, i.e., the energy consumed computation.

To obtain the energy consumption, we must observe Eq. 1, where E is the voltage supply of the measured device, I_j is the current input signal at a certain moment, T is the sample period of the input current signal and k is the number of samples along the time.

$$W[J] = \int_1^2 P\mathrm{d}t = \int_1^2 E(t) \cdot i(t)\mathrm{d}t = E \cdot T \cdot \sum_{j=1}^{k} I_j \tag{1}$$

Then, in Eq. 2 we compute the voltage measured by the ADC (V_{ADC}) where ADCbits is the number of bits of the ADC (12 bits in our case), result is the output value of the ADC, V_{refp} is the maximum voltage value of the ADC and V_{refn} is the minimum voltage value of the ADC. V_{ADC} coincides with the current measured by the µCurrent device ($I_{\mu\mathrm{Current}}$).

$$I_{\mu\mathrm{Current}} = V_{\mathrm{ADC}} = \frac{\mathrm{result} \cdot (V_{\mathrm{refp}} - V_{\mathrm{refn}})}{2^{\mathrm{ADCbits}}} + V_{\mathrm{refn}} \tag{2}$$

Since I_j is equal to $I_{\mu\mathrm{Current}}$, $\forall\, j \in (1, k)$ I_j coincides with each sample taken by the µCurrent, so we can define a system of equations with Eqs. 1 and 2 resulting in Eq. 3:

$$W[J] = E \cdot T \cdot \sum_{j=1}^{k} \left[\frac{\mathrm{result}_j \cdot (V_{\mathrm{refp}} - V_{\mathrm{refn}})}{2^{\mathrm{ADCbits}}} + V_{\mathrm{refn}} \right] = \frac{E}{f} \cdot \left[k \cdot V_{\mathrm{refn}} + \frac{V_{\mathrm{refp}} - V_{\mathrm{refn}}}{2^{\mathrm{ADCbits}}} \cdot \sum_{j=1}^{k} \mathrm{result}_j \right] \tag{3}$$

All variables of Eq. 3 are constant except the variable result. This variable varies depending on the I_j value. Therefore, we can split this equation in the equation system presented in Eq. 4, where the independent variable x is the accumulation of result values k times and C_1 and C_2 are constant. This system of equations can be handled as two computational blocks: 4a and 4b.

$$\begin{cases} x = \sum_{j=1}^{k} result_j & \text{(a)} \\ W[J] = C_1 + C_2 \cdot x & \text{(b)} \end{cases} \tag{4}$$

The computational block 4a is processed in the LPC824 (lines 39 and 42 of Algorithm 1) and the computational block 4b is processed in the PC. Thus, the data processing has two computational blocks in order to reduce the computational effort in the LPC824, while the PC shows the energy consumption measure. We have decided to do this separation of computational blocks since we can maximize the sample rate minimizing the LPC824 processing time. This is possible because sampling and data processing are done in parallel.

With respect to the computational block 4a, the result values are stored in the *BUFFER0* and *BUFFER1* by the data acquisition logic. Then, the LPC824 calculates x accumulating the result values. After that, the LPC824 sends the result of x to the PC through the UART. These operations are cyclical.

With respect to the computational block 4b, when the PC receives a new data, it multiplies x by C_2 and adds C_1. These operations are cyclical. Therefore, the PC calculates the energy consumption along the time using Eq. 5 where n_{sends} is the number of the transmissions done to the PC.

$$W_{\text{total}}[J] = \sum_{i=1}^{n_{\text{sends}}} W[J] \tag{5}$$

This data processing has limitations and it needs an experimental tuning. These limitations are described in Sect. 9 and the experimental tuning is described in Sect. 10.1. k is limited by the overflow problems of the x summation implementation. Also, k needs to be calculated in order to make the minimum number of transmissions to the PC, avoiding unnecessary transmission and maximizing the data processing speed. Finally, the maximum sample rate can be calculated depending on the temporal requirements of the Algorithm 1.

9 Setting the parameters of the approach

This section presents the limitations and temporal properties of the proposed approach that are inherent to the hardware on which it is implemented.

First, we must take care that the data processing depends on the summation of the result values k times (as in Eq. 4a). We have to take into account the size of the variable where the summation of the *result* is saved to avoid falling in overflow issues. For instance, if this variable has n bits of an unsigned int variable and the ADC measures in all the range, we can obtain the maximum number of samples until the buffer overflows (k_{max}) using Eq. 6.

$$2^n - 1 \leq \sum_{i=1}^{k_{\text{max}}} \left(2^{12} - 1\right); \quad k_{\text{max}} = \frac{2^n - 1}{2^{12} - 1} \tag{6}$$

The variable that stores the summation of all the result values must keep data until the next transmission to the PC. Then, this variable is sent to the PC and cleaned, allowing to store new data. Therefore, the k_{max} variable is the maximum number of samples until a data transmission.

The maximum number of filled buffers of *BUFFER0* and *BUFFER1* until transmit data to the PC (*n_buffers*) is limited by k_{max} and the size of the buffers (*buffer_size*). Thus, *n_buffers* is the nearest lower multiple obtained in Eq. 7. *n_buffers* is the variable specified on line 34 of Algorithm 1.

$$\mathit{buffer\_size} \cdot \mathit{n\_buffers} \leq k_{\text{max}}; \quad \mathit{n\_buffers} = \frac{2^n - 1}{\mathit{buffer\_size} \cdot (2^{12} - 1)} \tag{7}$$

Once *n_buffers* is determined, the final number of samples between transmissions to the PC (k_{maxreal}) is fixed by Eq. 8.

$$k_{\text{maxreal}} = \mathit{buffer\_size} \cdot \mathit{n\_buffers} \tag{8}$$

Besides that, it is worth mentioning that there are two important time constraints: the time to fill a buffer (t_{fill}) and the time to process the buffer (t_{process}). Each of the two times corresponds to a parallel task and both times must be respected for the correct operation of the technique.

The fill of the buffers never stops since we are interested in sampling data on a continuous basis. In this regard, Eq. 9 shows the temporal limitations. When the data processing ends processing a buffer, it waits until the other buffer is filled. Therefore, t_{process} is limited to t_{fill}. Thus, t_{fill} is the longest time between both times to ensure the continuous sampling process. In addition, the approach sends the information to the PC once in a while and therefore we must take into account the transmission time (t_{send}).

$$t_{\text{process}} + t_{\text{send}} \leq t_{\text{fill}} \tag{9}$$

To obtain the t_{process} we can experimentally find it and to obtain t_{send} we can calculate it, since the transmission protocol (UART) is well known. Therefore, we can obtain the maximum sample rate using Eq. 10, where f_{max} is the maximum sample rate to fill a buffer.

$$f_{\text{max}} = \frac{buffer\_size}{t_{\text{process}} + t_{\text{send}}} \tag{10}$$

Finally, the time between PC transmissions ($t_{\text{betweensends}}$) corresponds to Eq. 11 where t_{process} is the time of the last processed buffer and t_{send} is the transmission time.

$$t_{\text{betweensends}} = \frac{k}{f_{\text{max}}} + t_{\text{process}} + t_{\text{send}} \tag{11}$$

10 Experimental

The aim of this section is to validate our approach by running the data acquisition logic and the data processing in the selected hardware. First, the time settings and the values of the variables are obtained in Sect. 10.1 using the equations of Sect. 9. Then, in Sect. 10.2, the energy consumption of some basic circuits is measured in order to know the reliability of the technique. Finally, in Sect. 10.3 we assess the energy consumption measurement of a WSN mote.

10.1 Commissioning on experimental tuning of the approach

Next, the size set is defined. We will use the full ADC range, that is, henceforth from 0 (corresponding to 0 V) to 4095 (corresponding to 3.3 V). The variable size where the accumulation of the result values is stored has 64 bits for an unsigned int (n is equal to 64). Finally, the size of each buffer is set up to 1600 × 2 bytes since we save 1600 values of the ADC register and this register has 2 bytes (*buffer_size* is equal to 1600). We do not use the maximum SRAM size since there are other program variables saved in the SRAM and is sufficient to our approach.

We also need to determine the *n_buffers* value applying Eq. 7 and the k_{maxreal} value applying Eq. 8. In our experiments, *n_buffers* is equal to $2.81543713 \times 10^{12}$ and k_{maxreal} is equal to $4.504699407 \times 10^{15}$.

In order to obtain the maximum sample rate, first we calculate t_{process}. For this purpose, we set up the approach at a medium sample rate (in the SCTimer of the pipeline processing) of 95 KSPS, Then, we launch a LPC824 counter at the maximum allowed frequency (6 MHz) using the internal clock. We read two times the counter: before and after the data processing observing the changes in the counter value due to the data processing time. The result obtained is t_{process} is equal to 6.16 µs.

Then, we fix the UART baud rate at the maximum value of the protocol: 115200 BPS (Bits Per Second), and therefore, our data frame (t_{send}) takes 572.92 µs according to the UART protocol defined by Michael [27].

Finally, we calculate the maximum sample rate applying Eq. 10. $f_{\max}$ has a high value (order of GHz) that our system will never be able to reach since the maximum nominal sample rate of our hardware ADC is 1.2 Msamples/s. Then, the theoretical maximum sampling rate is limited by hardware.

10.2 Energy measure of basic circuits

In this section we have measured the energy consumption of some basic circuits in order to verify our approach. The consumption of these basic circuits is well-known and the difficulty of measuring them lies in another field of application. Thus, we have compared our approach with the theoretical consumption of these basic circuits in order to understand the errors produced in our measures.

We tested our approach in two types of basic circuits. The first circuit is an R circuit and the second one is a C discharge circuit. We have done tests for 1K and 1K8 resistances and for a 470 µF capacitor. The resistances are 1/4 w ± 5% and the capacitor is electrolytic ± 20%.

In order to measure the energy consumption using our approach, we have built a prototype board that contains all the hardware required. This board has the following values for Eq. 3 and for the algorithm: V_{refp} to 3.3 V, V_{refn} to 0 V, a frequency of 95 KSPS, *buffer_size* to 1600 and *n_buffers* to 18,000.

Each resistance circuit is continuously supplied with 2 V. The basic circuit composed of the capacitor is not powered; the capacitor is introduced charged at 2 V and it is discharged over time. At the same time, the prototype board is connected in series with each basic circuit measuring the energy consumption along 1.26 h.

Table 2 summarizes the energy consumption results. The first row shows the theoretical energy consumption in Joules (J) (Wtheoretical). The second row shows the energy consumption in J measured in our experiments (Wmeasured). The third row shows the error between Wtheoretical and Wmeasured energy consumption.

The error between Wtheoretical and Wmeasured is calculated applying Eq. 12. In this direction, the error in the $R = 1k$ circuit is 4.76%, the error in the $R = 1k8$ circuit is 3.27% and the error in the C circuit is unpredictable since when C is discharged, our

Table 2 Energy consumption experiments using our approach and theoretical methods for 1.26 h and voltage supply equal to 2 V

	$R = 1$**K**	$R = 1$**K8**	$C = 470$ **µF**
Wtheoretical [J]	18.144	10.08	0.00094
Wmeasured [J]	17.28	9.75	0.07
Error [%]	4.76	3.27	N/A

method measures leakage currents and noise. With these results we can conclude that the initial error is less than 5%.

$$\text{Error (\%)} = \frac{\text{Wtheoretical-Wmeasured}}{\text{Wtheoretical}} \times 100 \qquad (12)$$

10.3 Wireless sensor network consumption

Our technique was used to assess the energy consumption of a WSN ZigBee mote. This mote is a prototype composed of a XB24-Z7WIT-004 radio transceiver and a LPC824 microcontroller. In this regard, we compared the energy consumption of the mote when it implements a duty-cycling algorithm to reduce the energy consumption (LOKA) [37] and without it. The results are shown in Fig. 4. This figure is plotted taking into account that the mote was activated for a certain period of time and periodically it sends information to others motes.

The collected measures are divided into two experiments: without LOKA and with LOKA. In the first one, all the electronics components are in active mode. In the second one, the integrated algorithm switches the radio module on/off and it changes the power mode of the electronic components to reduce the energy used.

In the experiments, the mote sends periodically 64 bytes of data, allowing us to analyze the energy consumption for a specified operation time. By one side, the mote without LOKA was tested in 360 experiments divided in blocks of 30 measures. By the other side, the prototype mote with LOKA was tested in 60 experiments divided in blocks of 5 measures. The energy consumption profile obtained shows that the standard deviation is small so we considered that the number of samples was enough to support our conclusions.

For both experiments, we considered different values for the transmit interval time and the operation time as can be seen in Fig. 4.

- Discrete values of transmit data interval: [10, 30, 60] (s). These values correspond with the rectangle placed in the upper left corner and with each line plotted.
- Discrete values of operation time: [60, 300, 600, 900] (s). These values correspond to the abscissa axis.

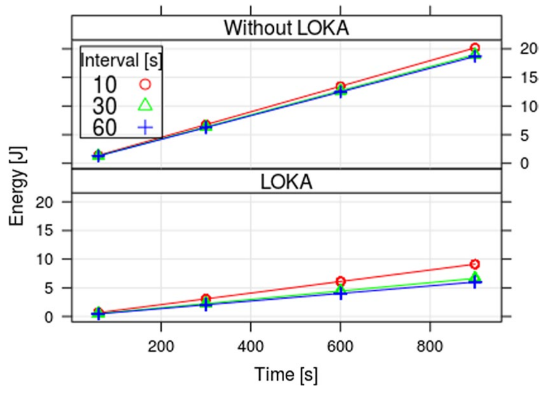

Fig. 4 Energy consumption measure of a ZigBee mote using the LOKA algorithm [37] and without it

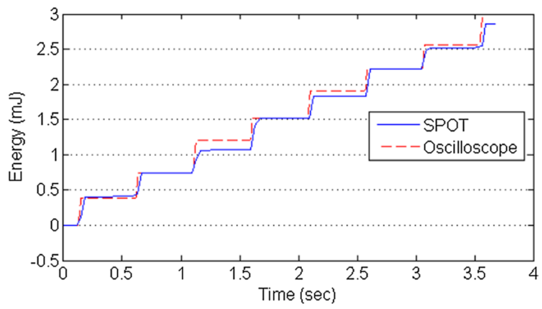

Fig. 5 Energy consumption measure of a TelosB mote using the Jiang et al. method (SPOT) with a 2 Hz duty-cycling algorithm [18]

Table 3 Comparison of our approach with the approach of Jiang et al. [18]

Time running the 0.1 Hz duty-cycling algorithm (s)	Jiang et al.	Our approach	
	TelosB (mJ)	ZigBee without LOKA algorithm (mJ)	ZigBee with LOKA algorithm (mJ)
60	2.4	1.38	0.65
300	12	6.75	3.07
600	24	13.46	6.10
900	36	20.17	9.12

Figure 4 shows that the energy consumption measured in joules is reduced along the time using the LOKA algorithm. The energy consumption results provided were similar to those obtained in the state of art [7, 12, 29].

The work of Jiang et al. [18] is quite similar to our approach. In this sense, they designed a method to measure energy consumption in WSN motes through a comparator and a counter, measuring the energy consumption of a *TelosB* WSN mote to verify their design. They analyzed the WSN mote running at slightly higher than 2 Hz duty-cycling algorithm and assessing it with an oscilloscope.

Comparing the work of Jiang et al. with our approach, in Fig. 5 we can observe that they obtain an approximation of a linear progression between 0 and 400 s composed of small energy consumption steps. Their approach cannot detect the energy consumption in sleep mode and therefore each step stays constant when the WSN mote is in sleep mode. On the contrary, our approach measures the energy consumption of the sleep state and adds it to the linear progression.

Moreover, we can estimate the energy consumption that they would measure if they ran a 0.1 Hz algorithm sending a message each 10 s. This is possible taking into account that each step in Fig. 5 corresponds to 0.4 mJ and it will only change the number of steps in a period of time: one energy step (of 0.4 mJ) every 10 s instead of one energy step every 0.5 s.

Table 3 shows the energy consumption for both approaches. The rows depict the time that the mote is running with a 0.1 Hz duty-cycling algorithm, the consumption of the TelosB mote, the consumption of our prototype mote with the LOKA algorithm and the consumption of our prototype mote without the LOKA algorithm.

We can see that the TelosB mote consumes more energy than our prototype, being the values in the same order of magnitude and the consumption between measures are constant for each mote such as in the original work of Jiang et al. The energy consumption of the TelosB mote is higher than the others since the microcontroller of the TelosB consumes more energy than the microcontroller of our mote prototype. Also, the energy consumption of the sensor motes depends on the data frame that the mote sends each transmission, the electronic devices connected to the mote, the electromagnetic noise, the channel of the transmission, the collision with other radio waves, etc.

11 Results

Not applicable. We do not need statistical analysis of appropriate, relative and/or absolute risks or risk reductions, and confidence intervals.

12 Discussion

Characterizing the energy consumption of electronic devices is a crucial factor when designing them. Specifically, if we want to know the consumption of an electronic device when the current input is random (it does not follow a pattern behavior), we need to empirically analyze it. In this regard, properly profiling the device's energy consumption is required to sample the input current signal as fast as possible in order to reduce data loss. With the available hardware in the market (without considering costly solutions), the sampling speed often is not enough and some techniques are needed for improving the sample rate.

The contribution of the paper is an algorithm that uses two DMA channels, a SCT-timer and an ADC of a microcontroller to deploy the double-buffered mechanism while improving the computational requirements of the data processing for measuring the energy consumption. With this technique the error is less than 5% in our experiments and the improved acquisition times are close to the nominal ADC times.

13 Conclusions and future work

In this paper we presented an energy measurement technique capable of greatly improving the sampling rate while processing the gathered data in parallel. The technique continuously measures the energy consumption using a double buffer in a low-cost hardware, a data acquisition logic and a data processing algorithm for maximizing the sample rate while reducing the computational load of the sampling hardware. This particular profiling technique is of relevant application to low consumption devices such as WSN motes; our approach is able to analyze the energy consumption of the motes in all the states, including the sleep mode.

In future works we will test the newest magnetoresistive sensors to understand its impact on the proposed technique.

Abbreviations

DAQ: Data acquisition; RISC: Reduced instruction set computer; ARM: Advanced RISC machine; ADC: Analog-to-digital converters; DMA: Direct memory access; WSN: Wireless sensors networks; DSP: Digital signal processors; RAM: Random access memory; CPU: Central processing unit; DC: Direct current; PCB: Prototype circuit board; JTAG/SWD: Joint test action group and serial wire debug; FPGA: Field-programmable gate array; THD: Total harmonic distortion; SRAM: Static random access memory; USB: Universal serial bus; SCTimer: State-configurable timer; UART: Universal asynchronous receiver–transmitter; IRQ: Interrupt request.

Authors' contributions
All authors equally contributed to the work. All authors read and approved the final manuscript.

Funding
This work is funded by the JCLM project SBPLY/19/180501/000024 and the Spanish Ministry of Science and Innovation project PID2019-109891RB-I00, both under the European Regional Development Fund (FEDER).

Availability of data and materials
Data generated or analyzed during the current study will be made publicly available on the GitHub of the group (https://github.com/ISG-UAH).

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 30 October 2020 Accepted: 11 August 2021
Published online: 28 August 2021

References

1. J. Andersen, M.T. Hansen, Energy bucket: a tool for power profiling and debugging of sensor nodes, in *Proceedings of the 3rd International Conference on Sensor Technologies and Applications* (Glyfada, Athens, Greece, 2009)
2. M.N. Baibich, J.M. Broto, A. Fert, F.N. Van Dau, F. Petroff, P. Etienne, G. Creuzet, A. Friederich, J. Chazelas, Giant magnetoresistance of (001)Fe/(001)Cr magnetic superlattices. Phys. Rev. Lett. **61**, 2472–2475 (1988)
3. R. Boll, K.J. Overshott et al., *Sensors, Magnetic Sensors*, vol. 5 (Wiley, Hoboken, 2008)
4. E. Boni, L. Bassi, A. Dallai, V. Meacci, A. Ramalli, M. Scaringella, F. Guidi, S. Ricci, P. Tortoli, Architecture of an ultrasound system for continuous real-time high frame rate imaging. IEEE Trans. Ultrason. Ferroelectr. Freq. Control **64**(9), 1276–1284 (2017)
5. E. Casilari, J.M. Cano-García, G. Campos-Garrido, Modeling of current consumption in 802.15. 4/ZigBee sensor motes. Sensors **10**(6), 5443–5468 (2010)
6. A. Castro, J.F. Martínez-Osuna, R. Michel, M. Escoto-Rodríguez, S.H. Bullock, A. Cueva, E. López-Reyes, J. Reimer, M. Salazar, S. Villarreal et al., A low-cost modular data-acquisition system for monitoring biometeorological variables. Comput. Electron. Agric. **141**, 357–371 (2017)
7. Crossbow Technology, *I. Mica2 Datasheet* (2017). http://www.xbow.com/
8. S.S.S.R. Depuru, L. Wang, V. Devabhaktuni, Smart meters for power grid: challenges, issues, advantages and status. Renew. Sustain. Energy Rev. **15**(6), 2736–2742 (2011)
9. F. Dettmann, U. Loreit, Sensor assembly for measuring current as a function of magnetic field gradient, US Patent 5,621,377 (1997)
10. M. Di Paolo Emilio, *Data Acquisition Systems: From Fundamentals to Applied Design*, vol. 1 (Springer, Berlin, 2013)
11. G. Dimitriou, P. Kikiras, G.I. Stamoulis, I. Avaritsiotis, A tool for calculating energy consumption in wireless sensor networks, in *Proceedings of the 10th Panhellenic Conference on Informatics (PCI)* (Volas, Greece, 2005)
12. H. Dubois-Ferrière, L. Fabre, R. Meier, P. Metrailler, Tinynode: a comprehensive platform for wireless sensor network applications, in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks* (Nashville, TN, USA, 2006)
13. P. Dutta, M. Feldmeier, J. Paradiso, D. Culler, Energy metering for free: augmenting switching regulators for real-time monitoring, in *Proceedings of the IEEE 7th International Conference on Information Processing in Sensor Networks (IPSN)* (Cambridge, Massachusetts, 2008)
14. C. Gay, S. Bracker, The e769 multiprocessor based data acquisition system. IEEE Trans. Nucl. Sci. **34**(4), 870–872 (1987)
15. S. Geng, C. Liu, J. Wang, L. Hou, Y. Yuan, Accurate low-current measurement circuit for multimeters and oscillograps. Indones. J. Electr. Eng. Comput. Sci. **12**(5), 3713–3718 (2014)
16. IAR Systems, *I-Jet User Guide* (2012)
17. IAR Systems, *I-Jet and i-Scope* (2018). www.iar.com/iar-embedded-workbench/power-debugging/
18. X. Jiang, P. Dutta, D. Culler, I. Stoica, Micro power meter for energy monitoring of wireless sensor networks at scale. in *Proceedings of the 6th International Conference on Information Processing in Sensor Networks* (Cambridge, MS, USA, 2007)
19. S.L. Jie, High-speed data transceiver system based on DDR2 SDRAM ping-pong double buffering. Chin. J. Electron Devices **38**(3), 12 (2015)
20. D. Kim, R. Managuli, Y. Kim, Data cache and direct memory access in programming mediaprocessors. IEEE Micro **21**(4), 33–42 (2001)
21. K. Le Phan, H. Boeve, F. Vanhelmont, T. Ikkink, F. De Jong, H. De Wilde, Tunnel magnetoresistive current sensors for IC testing. Sens. Actuators A Phys. **129**(1–2), 69–74 (2006)
22. J. Lenz, S. Edelstein, Magnetic sensors and their applications. IEEE Sens. J. **6**(3), 631–649 (2006)
23. F.L. Lewis, Wireless sensor networks, in *Smart Environments: Technologies, Protocols, and Applications*, vol. 11, ed. by D.J. Cook, S.K. Das (Wiley, Hoboken, 2004), pp. 11–46. https://doi.org/10.1002/047168659X
24. S. Li, J.R. Luo, Y.C. Wu, G.M. Li, F. Wang, Y. Wang, Continuous and real-time data acquisition embedded system for east. Trans. Nucl. Sci. **57**(2), 696–699 (2010)

25. J. Lin, Z. Xu, A. Nukada, N. Maruyama, S. Matsuoka, Optimizations of two compute-bound scientific kernels on the sw26010 many-core processor, in *Proceedings of the IEEE 46th International Conference on Parallel Processing (ICPP), 46th (August 2017)* (Bristol, UK, 2017)
26. N. McNeill, N.K. Gupta, W.G. Armstrong, Active current transformer circuits for low distortion sensing in switched mode power converters. IEEE Trans. Power Electron. **19**(4), 908–917 (2004)
27. M.S. Michael, Universal asynchronous receiver/transmitter, March 1993. US Patent 5,199,105
28. R. Moghe, A.R. Iyer, F.C. Lambert, D.M. Divan, A low-cost wireless voltage sensor for monitoring MV/HV utility assets. IEEE Trans. Smart Grid **5**(4), 2002–2009 (2014)
29. J. Polastre, R. Szewczyk, D. Culler, Telos: enabling ultra-low power wireless research, in *Proceedings of the 4th International Symposium on Information processing in sensor networks* (Los Angeles, California, USA, 2005)
30. P. Poulichet, F. Costa, É. Labouré, A new high-current large-bandwidth dc active current probe for power electronics measurements. IEEE Trans. Ind. Electron. **52**(1), 243–254 (2005)
31. W. Ray, C. Hewson, High performance Rogowski current transducers, in *Proceedings of the IEEE 35th IAS Annual Meeting and World Conference on Industrial Applications of Electrical Energy* (Roma, Italy, 2000)
32. D. Rossi, I. Loi, G. Haugou, L. Benini, Ultra-low-latency lightweight DMA for tightly coupled multi-core clusters, in *Proceedings of the 11th ACM International Conference on Computing Frontiers* (Association for Computing Machinery, 2014)
33. J.C. Sancho, D.J. Kerbyson, Analysis of double buffering on two different multicore architectures: quad-core opteron and the cell-be, in *Proceedings of the 22nd IEEE International Symposium on Parallel and Distributed Processing* (Miami, USA, 2008)
34. M.I. Shaik, Design amp; implementation of arm based data acquisition system, in *Proceedings of the IEEE International Conference on Electronics, Communication and Computing Technologies* (Tamil Nadu, India, 2011)
35. X. Tan, X. Shen, X. Ye, D. Wang, D. Fan, L. Zhang, W. Li, Z. Zhang, Z. Tang, A non-stop double buffering mechanism for dataflow architecture. J. Comput. Sci. Technol. **33**(1), 145–157 (2018)
36. Texas Instruments, *Msp430 Advanced Power Optimizations: Ulp Advisor Software and Energytrace Technology* (2014)
37. D. Vaquerizo-Hdez, P. Muñoz, M.D. R-Moreno, D.F. Barrero, A low power consumption algorithm for efficient energy consumption in zigbee motes. Sensors **17**(10), 2179 (2017)
38. E.G. Vidal, D.R. Muñoz, S.I.R. Arias, J.S. Moreno, S. Cardoso, R. Ferreira, P. Freitas, Electronic energy meter based on a tunnel magnetoresistive effect (TMR) current sensor. Materials **10**(10), 1134 (2017)
39. C. Wang, J. Zhang, F. Yang, C. Du, One-shot method for purification of multiple natural amelogenin isoforms. J. Mater. Sci. Technol. **15**, 1–7 (2017)
40. S. Ziegler, R.C. Woodward, H.H.-C. Iu, L.J. Borle, Current sensing techniques: a review. IEEE Sens. J. **9**(4), 354–376 (2009)
41. C. Zinner, W. Kubinger, ROS-DMA: a DMA double buffering method for embedded image processing with resource optimized slicing, in *Proceedings of the IEEE 12th International Conference on Real-Time and Embedded Technology and Applications* (San Jose, CA, USA, 2006)

Publisher's Note