Received July 29, 2021, accepted September 20, 2021, date of publication September 23, 2021, date of current version October 7, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3115440

A Novel Low-Power Time Synchronization Algorithm Based on a Fractional Approach for Wireless Body Area Networks

GIUSEPPE COVIELLO[1], (Member, IEEE), GIANFRANCO AVITABILE[1], (Senior Member, IEEE), ANTONELLO FLORIO[1], (Student Member, IEEE), CLAUDIO TALARICO[2], (Senior Member, IEEE), AND JANET M. WANG-ROVEDA[3]

[1]Department of Electrical and Information Engineering, Polytechnic University of Bari, 70125 Bari, Italy
[2]Department of Electrical and Computer Engineering, Gonzaga University, Spokane, WA 99258, USA
[3]Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85721, USA

Corresponding author: Giuseppe Coviello (giuseppe.coviello@poliba.it)

The work of Giuseppe Coviello was supported by the Italian Ministry of Education, University, and Research through the European National Operational Program (PON) Project Attraction and International Mobility (AIM), activity 1, line 1 under Grant AIM1892824.

ABSTRACT Time synchronization is a topic of interest for any distributed system and it is of particular relevance in Wireless Sensor Networks (WSN), especially when it is necessary to keep a strong level of time synchronization among the data coming from different nodes, which are then employed to perform *a posteriori* data-fusion and data-merging operations. A special case of WSN is constituted by Wireless Body Area Networks (WBAN). The paper introduces the Fractional Low-power time Synchronization Algorithm (FLSA), a lightweight and ultra-low-power time synchronization algorithm conceived for Wireless Body Area Networks. The core of the proposed approach is the fractional-time concept, borrowed from Phase-Locked Loops theory, that allows achieving fine timer corrections. Moreover, an heuristic routine managing the on/off switching of the radio section of the device allows to dramatically decrease the power consumption. The mathematical discussion, along with a set of experiments is presented, proving the benefits associated with the proposed algorithm.

INDEX TERMS Wireless body area networks, wireless sensor networks, time dissemination, time synchronization, wireless application protocol.

I. INTRODUCTION

Sensing specific physical environment variables is a vital requisite for many applications of interest for the scientific community. The management of this important task benefitted from the advent of several new efficient communication techniques including the introduction of the WSN paradigm. A WSN is a collection of remotely connected sensors and actuators, called nodes, devoted to sense, measure, quantify, and further process a set of physical variables or, somehow, interact with them [1]. Many technologies have been developed to design the sensors. Among these, the advent of Micro-Electrical-Mechanical Systems (MEMS) allowed designing power-efficient and low-cost nodes. MEMS integrate on the same substrate a set of electronic, optical, and mechanical components. The main advantages determined by the high level of integration of MEMS technology are low power consumption, reliability, robustness and low production costs [2]. The WSN main limitations derive from the communication medium, that is the radio channel, and from the computational power limits of the nodes. In particular, this latter characteristic is a consequence of the power efficiency demanded from most of the operating scenarios, in which the nodes are designed to be battery powered.

A particular type of WSN is the WBAN. A WBAN is a network in which sensors and actuators are placed along or implanted in a body and the communication is performed through the radio medium [3]. WBANs can have both medical and non-medical applications [4]. In the first case, the objective is to reduce the need for hospitalization of patients for

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wei.

monitoring purposes, therefore the tasks of the WBANs are to measure the vital signs and report them to the medical staff. In the second case, WBANs can have multiple possible tasks in entertainment, gaming, consumer electronics, non-medical emergency handling (typically interfacing with outer sensor networks) and emotion detection [4].

WSNs and WBANs are important examples of distributed systems. Clock synchronization is a critical issue in every distributed environment [5]. In fact, in this scenario each node is governed by its own physical clock and, given the fact that each of them can be subject to some artifacts, there is the need for a common reference in order to avoid possible time mismatches. Time synchronization is important for many reasons in WSNs. For example, if we aim to merge and correlate data coming from different sensing sources, it is important to ensure their time coherency. A common example where time consistency is essential are fault-recovery applications.

Time synchronization accuracy varies as a function of the different applications. During the years, many synchronization protocols and algorithms have been developed to solve the clock coordination problem in a distributed environment [6]–[11].

However, in the case of WSNs and WBANs, their specific constraints do not allow to employ the classical techniques. In fact, the limited power available requires protocols that reduce at its minimum the number of messages associated to time coordination, and to the processing time and effort associated to process them. Moreover, since the radio channel provides an unreliable transmission medium, it is mandatory to employ protocols that are as simple as possible, even at the cost of losing packets, due to the presence of interferers or absorbers.

In this paper, we introduce a novel time synchronization algorithm for WBANs. The algorithm is capable of both time offset and time skew compensation. In particular, as it will be explained in the next sections, the proposed approach is based on a novel fractional-time correction technique relying on a dual modulus division, and an heuristic routine that balances the power-consumption and the precision of the system timer according to thresholds that are customizable with respect to the application.

The paper is organized as follows: Section II describes the background and the analysis of other techniques present in literature. In particular, it focuses on the definition and modeling of clock artifacts, the characterization of delays in message exchanging and the most known and employed algorithms and protocols for time synchronization in WSNs, with a particular emphasis on the WBAN case. Section III describes the theory behind the classic integer clock implementation. Section IV introduces the main motivations for which the described approach was devised. Then, in Section V, the algorithm theory and operations are presented in detail. Section VI describes some experiments conducted for validating the approach for both a precision and power efficiency. Finally, conclusions close this work.

II. THEORETICAL BACKGROUND AND RELATED WORKS

In this section, after having briefly recalled WBAN design requirements and topologies, we will introduce some basic background concepts to understand how time synchronization algorithms and protocols operate.

Later on, we will discuss the advantages and the disadvantages of some of the most common algorithms and approaches for time synchronization.

A. WIRELESS BODY AREA NETWORKS

The classical definition of WBAN has been already presented in the introduction [3]. WBANs have been subject to standardization from the IEEE 802.15 working group and their main technical requirements are specified in the IEEE 802.15.16 standard draft [12]. Critical issues associated to the development of a WBAN are [4] the minimization of the communication delay in the network and the network itself maximum throughput rate. Also, it is important to guarantee good network lifetime (i.e., the resiliency to network failures must be taken into account), and minimum power consumption. The nodes in a WBAN are usually battery powered, for portability reasons.

The IEEE 802.15.16 draft considers the star topology as the main network topology for a WBAN [12], in other words, it is the central node that coordinates the entire network. Star topology can be of the single or multi-hop type. However, to guarantee resiliency to network failures, and facilitate sensors/actuators addition or removal, the single-hop star topology is more efficient. This paper will focus on the single-hop topology.

B. A MODEL FOR LOW-COST DEVICES CLOCK

Usually in low-cost devices, the clock is generated by crystal oscillators that ensure reasonably stable operations with the benefit of reduced costs.

Let us define $\chi_i(t)$ a function that allows to determine the time value for the device i at the time instant t, starting from the clock circuit operating at frequency f_{SRC}. In perfect conditions, for each node i:

$$\chi_i(t)\big|_{ideal} = \frac{\alpha(t)}{f_{\mathrm{SRC}}}, \quad \alpha \in \mathbb{R} \tag{1}$$

where $\alpha(t)$ is a proper scaling factor. In practice, real clocks experience some non-idealities, due to temperature variations [13], aging [14] and load capacitance mismatches [15]. In a distributed environment, these eventually lead to have different time values for each node. Because of the relativity of time computation, if we aim to model those artifacts we have to use a common reference clock value. Hence, if i is the generic node, equation (1) must be corrected as follows:

$$\chi_i(t) = \varepsilon_i(t) + \vartheta_i, \quad \forall i \tag{2}$$

being $\varepsilon_i(t) = \alpha_i(t)/(f^i_{\mathrm{SRC}})$ and ϑ_i being the initial offset. Therefore, if we suppose j to be the reference node,

$$\chi_j(t) = \varepsilon_j(t) + \vartheta_j \tag{3}$$

we have

$$\Delta\chi_{ij}(t) \triangleq \chi_i(t) - \chi_j(t) = \varepsilon_i(t) - \varepsilon_j(t) + \Delta\vartheta_{ij} \quad (4)$$

Because of its time invariant nature, $\Delta\vartheta_{ij}$ can be compensated in a calibration phase. Then $\forall t$ there is a non-static error due to the different crystal oscillation frequencies, commonly called skew. If we write $1/(f^i_{\mathrm{SRC}}) = t^i_{\mathrm{SRC}}$ and suppose $t^i_{SRC} = t^j_{\mathrm{SRC}} + e_{ij}(t)$, then

$$\Delta\chi_{ij}(t) = [\alpha_i(t) - \alpha_j(t)]\, t^j_{SRC} - \alpha_i e_{ij}(t) \quad (5)$$

with $e_{ij}(t)$ modeling the instantaneous time error due to frequency drifts. The implementation of a clock correction can be difficult and expensive at the hardware level, (expecially in terms of power consumption), and, thus, it is usually performed in software. To correct the drift, we can vary the proportionality coefficients. So, in order to correct i (i.e. to obtain $\Delta\chi_{ij}(t) = 0$) we should have

$$\alpha_i(t) = \frac{\alpha_j(t)\, t^j_{\mathrm{SRC}}}{t^j_{SRC} + e_{ij}(t)} = \frac{\alpha_j(t)}{1 + f^j_{\mathrm{SRC}}\, e_{ij}(t)} \quad (6)$$

C. ALGORITHMS AND PROTOCOLS FOR WSN TIME SYNCHRONIZATION

Being the WBAN a particular type of WSN, the protocols employed for time synchronization are usually inherited from this latter operative context. Different operative scenarios call for diverse protocols and approaches. Almost every protocol has a single node acting as the time reference for the whole WSN, therefore each node must be synchronized to its timing. We will refer to this node as the root-node, time coordinator node or time-reference node as in [16]. Note that, in the WBAN case, this node is, actually, the star-center node. We can so refer to this kind of communication paradigm as a master-slave approach, with the root-node acting as the master of the synchronization process and the other nodes being the slaves.

When aiming to perform data merging from the different sensors of the network, in many applications it is not necessary for the master's time to be absolutely correct (e.g. with respect to the UTC). In fact, what is relevant is that the nodes agree on a common time.

Depending on the quantities that they correct, clock correction and synchronization protocols and algorithms can be divided in offset-only correction algorithms and joint skew-offset correction algorithms. Offset-only solutions generally require a higher number of messages, because the time-variant term leads to the impossibility of a single correction. Hence, joint skew-offset solutions are eligible to be lower power-intensive techniques.

In those systems in which we expect a huge impact of frequency drifting, a greater complexity for the algorithms is required because it is necessary to predict how the drifting will evolve, in order to reduce the frequency of the synchronization messages. To do that, regression and machine learning techniques can be employed [17]–[19]. In very large WSNs, there are many approaches to simplify and speed up the synchronization phase. One example is presented in [20] in which the network is divided into broadcast subdomains and the synchronization message dissemination is performed thanks to the creation of a spanning tree. Spanning trees are also employed in [19] and [21] as a way to manage multi-hop. In this way it is also possible to manage time-coordinator failures.

For those scenarios in which the links between the nodes are not stable, a resistive-network inspired approach like the one proposed in [17] can be very helpful because of its flexibility. However, given the computational weight associated with the steps, the effort could be justified only for very large WSNs.

To further simplify the complexity of distributed protocols, another trend is furnished by overhearing-based protocols, such as [22], [23], in which neighbor nodes capture the synchronization packet during an *ad hoc* synchronization phase (usually pair-wise) of a couple of nodes.

In the wide literature currently available, some protocols are taken as a reference because of their large diffusion. One of those is the Reference Broadcast Algorithm (RBS) [18], used to synchronize a set of receivers to the coordinator. RBS does not employ dedicated synchronization messages and adopts relative time synchronization. Once the root-node has sent the `reference_packet`, the receivers log the local time of reception associated with that packet. Then, the receivers exchange their observation and, once they have collected the whole network observations, they average and reach a consensus on the average time of reception. Note that RBS is not optimized for low-power contexts. However, in [24], a revised RBS approach for low-power energy-harvested WSN is proposed, in which a linearly incremental switch-off time for the radio section is employed when the synchronization error decreases. Otherwise, the off-time is reduced referring to the slope of the interpolated error samples (or, equivalently, the "time points" received from the master).

The Timing synchronization Protocol for Sensor Networks (TPSN) [21] is a dynamic pair-wise synchronization approach in which synchronization-request senders are synchronized to receivers. It has two main phases. The first one is a *discovery* phase, initiated by the root node. At this stage, a broadcast message is sent by every node to establish a spanning tree hierarchy based on level number exchange. Then, in the *synchronization* phase, each node belonging to level i sends a `synchronization_pulse` to the one on level $i-1$ (assuming the root-node has level 0), which answers with an acknowledge (`ACK`) message.

Another example is constituted by the Flooding Time Synchronization Protocol (FTSP) [19]. FTSP synchronizes the sender time with multiple receivers using message timestamped at MAC layer. The approach is simple, i.e. the central node floods the entire network with a broadcast packet containing its local MAC time, so that each receiver logs the reception time and adjusts its own MAC time to it.

D. A MODEL FOR THE TIME DELAY IN MESSAGE EXCHANGE

The time elapsed for the synchronization message from when it is generated up to when it is processed at the receiver's side can be decomposed into 5 parts [19]:

- Send Time (t_S): time needed for the message to be prepared and sent to the network interface;
- Access Time (t_A): waiting time for the shared communication channel to be free;
- Transmission Time (t_{TX}): delay needed for the sender to put the message on the communication channel;
- Propagation Time (t_P): time needed for the electromagnetic signal to travel from the transmitter to the receiver;
- Reception Time (t_R): It is the dual of t_S, and it goes from receiving the synchronization message to the end of its processing.

The overall delay τ can be expressed as:

$$\tau = t_S + t_A + t_{TX} + t_P + t_R \tag{7}$$

Note that some of the contributions to the overall time delay have a random nature (i.e. t_S, t_A, t_R). However, this model can be simplified under some hypotheses.

As previously discussed, in time synchronization there is commonly a reference node and every node synchronizes to it.

Since we are assuming a star topology, we can consider that the central node uses messages of the broadcast type to communicate its time. In this specific case, it is possible not to consider t_S, t_A and t_{TX} for all the slaves. Since a single message is broadcast to all the nodes there is only a unique send-time to wait. Channel access time is also the same for every node, as a single t_A is needed to be waited before sending the synchronization message to all the other nodes. For what concerns t_{TX}, since it is related to the message length, for a broadcasted message it represents a quantity that it is equal for each slave. Therefore, the possible inter-slave delay expression reduces to:

$$\tau \approx t_R + t_P \tag{8}$$

If the considered network has a small spatial extension, as in the case of a WBAN, t_P can be neglected, because, considering the air as propagating medium ($c = 3 \cdot 10^8$ m/s), t_P ranges from 10 to 100 nanoseconds for a distance of $1 - 20$ m. Hence, we can accordingly simplify the expression of τ:

$$\tau \approx t_R \tag{9}$$

Thus, to further optimize the overall delay, τ, we have to reduce t_R. However, due to t_R random nature, a good reception queue management policy can also lead to significant performance improvements.

III. INTEGER CLOCK IMPLEMENTATION

Usually, in a low-cost system, the System Timer (ST) is supplied by a Real-Time Clock (RTC) module. For low-power applications, the RTC modules usually integrate a set of registers (counter register, capture/compare register, prescaling register) and a tick event generator, as shown in Fig. 1. The relationship between the frequency value of the clock source, f_{SRC}, and the increment rate f_{RTC} of the counter register R_{COUNT}, is:

$$f_{\mathrm{RTC}} = \frac{f_{\mathrm{SRC}}}{1 + R_{\mathrm{P}}} \tag{10}$$

where R_{P} is the value of the prescaler register.

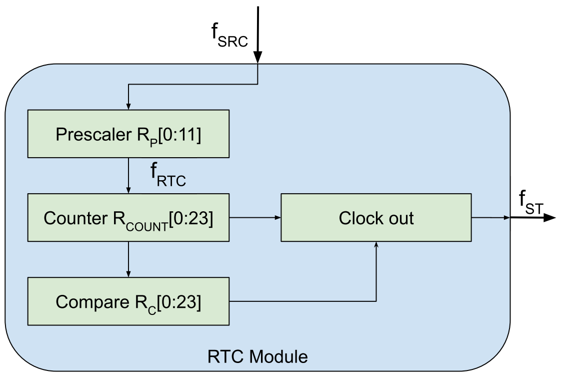

FIGURE 1. Typical ST implementation.

The easiest way to implement a ST is to set its resolution, t_{ST}, by properly programming the prescaler. In this case, the ST resolution is equal to that of the RTC, t_{RTC}:

$$t_{\mathrm{ST}} = t_{\mathrm{RTC}} = \frac{1}{f_{\mathrm{RTC}}} \tag{11}$$

The ST value can be directly obtained by multiplying the $\mathrm{R_C}$ register value and the ST resolution value:

$$\mathrm{ST} = t_{\mathrm{ST}}\,\mathrm{R_{COUNT}} = t_{\mathrm{RTC}}\,\mathrm{R_C} \tag{12}$$

The maximum value that can be stored in the ST, $\mathrm{ST_{MAX\_VALUE}}$, represents the only limit for such implementation:

$$\mathrm{ST_{MAX\_VALUE}} = t_{\mathrm{ST}}(2^{\mathrm{N}} - 1) = t_{\mathrm{RTC}}(2^{\mathrm{N}} - 1) \tag{13}$$

where N is the length in terms of bits of the counter register.

Commonly employed register lengths are 24 bits for the counter and 12 bits for the prescaler register. The used crystals operate at a frequency $f_{\mathrm{SRC}} = 32.768$ kHz [25]. As a consequence, t_{ST} is:

$$30.517\,\mu\mathrm{s} \leq t_{\mathrm{ST}} \leq 125\,\mathrm{ms} \tag{14}$$

so that the maximum ST value $\mathrm{ST_{MAX\_VALUE}}$ is such that:

$$512\,\mathrm{s} \leq \mathrm{ST_{MAX\_VALUE}} \leq 582.542\,\mathrm{hours} \tag{15}$$

Once the desired resolution has been set, the maximum value that the ST can manage without overflowing and resetting the register R_{COUNT} to zero can be computed. The dependence between resolution and maximum ST value can be overcome using the internal RTC module feature called `compare`. When the $\mathrm{R_{COUNT}}$ register value reaches the value stored in the comparison register, $\mathrm{R_C}$, an interrupt is issued. The interrupt routine manages some operations linked to the ST itself at a higher level. Under this condition, the ST frequency, f_{ST}, becomes:

$$f_{\mathrm{ST}} = \frac{f_{\mathrm{RTC}}}{\mathrm{R_C}} \tag{16}$$

where f_{ST} represents the ST refresh frequency and $\mathrm{R_C}$ is the value of the imposed comparison value. Note that the compare register length is usually the same as the length of the counter register.

The ST resolution is given by:

$$t_{\mathrm{ST}} = \frac{1}{f_{\mathrm{ST}}} = \frac{\mathrm{R_C}}{f_{\mathrm{RTC}}} \tag{17}$$

so that it is possible to find

$$t_{\mathrm{ST}} = \frac{\mathrm{R_C}(1+\mathrm{R_P})}{f_{\mathrm{SRC}}} \geq \frac{(1+\mathrm{R_P})}{f_{\mathrm{SRC}}} \tag{18}$$

Despite having decreased the ST resolution by a factor $\mathrm{R_C}$ with respect to the RTC resolution, it can still be varied in t_{RTC} steps, so the granularity of the ST continues to be very low. A practical example helps to gain a better insight. Let us assume a system employing a $f_{\mathrm{SRC}} = 32.768$ kHz RTC source and with a $f_{\mathrm{ST}} = 1$ kHz target ST frequency. A possible implementation that allows to successively refine in steps of the ST resolution, 30.517 μs, consists in setting the $\mathrm{R_P}$ to zero and the register $\mathrm{R_C}$ to:

$$\mathrm{R_C} = \left\lceil \frac{f_{\mathrm{RTC}}}{f_{\mathrm{ST}}} \right\rceil = \left\lceil \frac{32.768\,\mathrm{kHz}}{1\,\mathrm{kHz}} \right\rceil = \lceil 32.768 \rceil = 33 \tag{19}$$

The example illustrates that the approximation introduced starting with this setting is about 7.08 μs/ms. In the forthcoming section we will illustrate the effect of this choice on the overall system precision.

IV. MOTIVATIONS TO THE PROPOSED APPROACH

The most intuitive strategy to deploy in synchronized master-slave systems with a star topology is to calibrate the unit ST according to that of the master.

This strategy assumes that the slaves correct their ST and calibrate the whole system to keep a stable synchronization. The operation is performed by every single slave using the synchronization messages received by the master. In this case, the great advantage is that each slave autonomously decides whether to receive every synchronization packet turning the radio on or to skip some synchronization slots as a function of its capability to keep the synchronization with the master. The slaves can accelerate or decelerate their ST according to the needs and this is done by changing the value in the $\mathrm{R_C}$ register, that is, the number of ticks, t_{RTC}, determining the t_{ST}. Let us evaluate how the effects of a δ_{ST} variation in the t_{ST} affect the overall ST. δ_{ST} can be either a positive or a negative number, depending on the need to accelerate or decelerate the global ST and it is an integer multiple of t_{RTC}.

$$\delta_{\mathrm{ST}} = k t_{\mathrm{RTC}} \qquad k \in \mathbb{Z} \tag{20}$$

Let $\mathrm{T_M}$ be the repetition rate of the master synchronization message toward the slaves. The ST variation in each $\mathrm{T_M}$ period, $\Delta\mathrm{ST}$, is:

$$\Delta\mathrm{ST} = \frac{T_{\mathrm{M}}}{t_{\mathrm{ST}}} \delta_{\mathrm{ST}} \tag{21}$$

Thus, the ST variation is directly proportional to the repetition rate of the synchronization message, $\mathrm{T_M}$, and inversely proportional to the ST time resolution, t_{ST}. Using (16), we obtain:

$$\Delta\mathrm{ST} = \frac{k\,\mathrm{T_M}}{\mathrm{R_C}} \tag{22}$$

In an integer clock representation, the slave can only change the $\mathrm{R_C}$ value of a quantity k. Hence, the minimum insertable step during a synchronization period $\Delta\mathrm{ST_{MIN}}$ is obtained with $|k| = 1$ and it is given by

$$\Delta\mathrm{ST_{MIN}} = \frac{\mathrm{T_M}}{\mathrm{R_C}} \tag{23}$$

To clarify the concept, let us consider an example.

Let us choose to keep t_{RTC} as low as possible, that is equal to 30.517 μs. The results are reported in Table 1. Table 1 illustrates that the desired $\Delta\mathrm{ST_{MIN}}$ values can be reached by changing $\mathrm{T_M}$ and/or $\mathrm{R_C}$. This implementation does not allow obtaining all the possible values for $\Delta\mathrm{ST_{MIN}}$ since $\mathrm{R_C}$ is an integer value. This is a common problem in those contexts in which integer dividers are employed. To obtain also fractional values, one of the classical techniques is the dual-modulus division [26] that consists of employing two counters and two registers. In this way, if we call the two register values $\mathrm{R_C^1}$ and $\mathrm{R_C^2}$ we obtain:

$$\Delta\mathrm{ST_{MIN}}\Big|_{\mathrm{frac}} = \frac{\mathrm{T_M}}{\mathrm{R_C^1}\cdot\mathrm{I_1} + \mathrm{R_C^2}\cdot\mathrm{I_2}} \tag{24}$$

TABLE 1. ST granularity variation as a function of t_{RTC}, $\mathrm{R_C}$ and $\mathrm{T_M}$ parameters.

t_{RTC}	$\mathrm{R_C}$	$\mathrm{T_M}$	t_{ST}	$\Delta\mathrm{ST_{MIN}}$
30.517 µs	33	20 s	1 ms	606.06 ms
30.517 µs	33	4.024 s	1 ms	121.95 ms
30.517 µs	164	20 s	5 ms	121.95 ms

with $\{\mathrm{I}_i\}_{i=1,2}$ being the times each register value should be taken into account during the observation period $\mathrm{T_M}$. The ideal condition is obtained when

$$\Delta\mathrm{ST_{MIN}} = t_{\mathrm{ST}} \tag{25}$$

t_{ST} strictly depends on the application. Let us assume, for instance, an Inertial Measurement Unit (IMU) whose data need to be sampled at a $f_s = 200$ Hz rate that is, $t_s = 5$ ms. Let us further suppose the IMU is equipped with a RTC whose $f_{\mathrm{RTC}} = 32.768$ kHz. In this case, the t_{ST} must be either equal to the IMU sampling period, 5 ms, or, to gain some safety margin, to one of its submultiples, for

TABLE 2. Notation.

Name	Type	Description
$\mathrm{R_C^i}$	Integer positive	Number of ticks needed to form a t_{ST} for comparator $i \in \{\mathrm{A}, \mathrm{B}\}$
I_i	Integer positive	Number of times comparator i should be used $i \in \{A, B\}$
$\delta\mathrm{R}_\mathrm{C}^i$	Integer	Needed tick variation to form t_{ST} for comparator $i \in \{\mathrm{A}, \mathrm{B}\}$
$\delta\mathrm{I}_i$	Integer	Variation of the count of comparator i uses, $i \in \{\mathrm{A}, \mathrm{B}\}$
$\Delta\mathrm{T}(j)$	Real	Difference between the local timestamp for node j and that sent by the master
$\mathrm{T_M}$	Real positive	Repetition time in seconds for the master synchronization packet, i.e. slot length
$\Delta\mathrm{T_M}(j)$	Real	Equivalent accumulated time difference between the master and the node j in a single slot $\mathrm{T_M}$
$\Delta\mathrm{T}_{\mathrm{M},\{\min,\max\}}(j)$	Real	Minimum and maximum values of $\Delta\mathrm{T_M}(j)$ for node j
$\mathrm{N_M}$	Integer positive	Ideal number of ticks to be counted every $\mathrm{T_M}$
$\Delta\mathrm{N}$	Integer	Number of ticks to recover from lead/lag
$\Delta\mathrm{N_M}$	Integer	number of ticks to recover lead/lag in relation to each $\mathrm{T_M}$
$\mathrm{S_S}(j)$	Integer positive	Number of skipped synchronization slots by node j
$\mathrm{AVG}_{wd}(j)$	Real positive	Weighted average of delays for node j

example 1 ms. If we consider the ideal condition (25), given the expression (23), $\mathrm{T_M}$ cannot be greater than 820 ms and 33 ms for a ST resolution of 5 ms and 1 ms respectively (see the information in Table 1). Such values cause an excessive power consumption due to the intensive radio channel use and to the huge message exchange.

To ensure a reduced message exchange while retaining a high accuracy level, it is apparent that a new synchronization algorithm based on the fractional approach should be introduced.

V. ALGORITHM

The novel algorithm proposed is called FLSA and its core is based on four sub-routines, each one of them being in charge of a specific task related to the global procedure. A high-level diagram of the algorithm is shown in Fig. 2. The diagram emphasizes the interactions between the algorithm's sub-routines. The first sub-routine initializes the fractional timing process. The second one regulates the processing priorities of the messages received by each slave, making a distinction between the synchronization-related messages and other messages. The third sub-routine employs the data coming from the other subroutines to implement the slave time correction and synchronization to the master. The last sub-routine optimizes the power consumption through a time-slot skipping strategy.

A. FRACTIONAL TIMING FUNDAMENTALS

We already mentioned the possibility of improving the granularity of the minimum insertable ST increment by adopting a couple of registers/counters instead of a single one. In order to simplify the notation, we will now refer to the number of ticks rather than to time values. The required number of ticks to form a synchronization slot is:

$$\mathrm{N_M} = \left\lfloor \frac{\mathrm{T_M}}{\mathrm{t_{RTC}}} \right\rceil \tag{26}$$

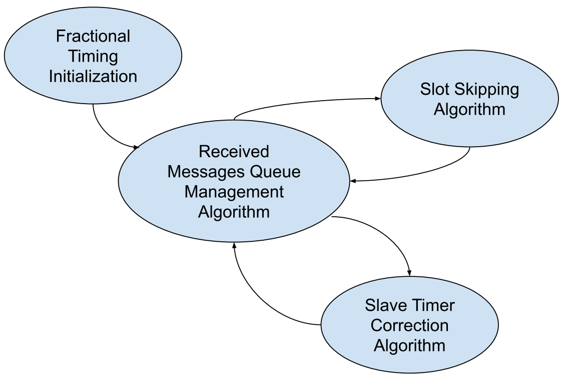
Fractional Timing Initialization
Slot Skipping Algorithm
Received Messages Queue Management Algorithm
Slave Timer Correction Algorithm

FIGURE 2. FLSA routines interaction diagram.

where $\lfloor\cdot\rceil$ represents the rounding at the closest integer number.
The first fundamental relationship is:

$$\mathrm{N_M} = \mathrm{R_C^A} \cdot \mathrm{I_A} + \mathrm{R_C^B} \cdot \mathrm{I_B} \tag{27}$$

and for any variation

$$\Delta\mathrm{N_M} = (\mathrm{R_C^A} + \delta\mathrm{R_C^A}) \cdot \delta\mathrm{I_A} + (\mathrm{R_C^B} + \delta\mathrm{R_C^B}) \cdot \delta\mathrm{I_B} \tag{28}$$

should be imposed.

B. FRACTIONAL TIMING INITIALIZATION

Given $\mathrm{T_M}$ and t_{ST}, the system of equations to be solved to evaluate the initialization values is:

$$\begin{cases} \mathrm{R_C^A} \cdot \mathrm{I_A} + \mathrm{R}_\mathrm{C}^B \cdot \mathrm{I_B} = \mathrm{N_M} \\ \mathrm{I_A} + \mathrm{I_B} = \left\lfloor \dfrac{\mathrm{T_M}}{t_{\mathrm{ST}}} \right\rceil \end{cases} \tag{29}$$

If we use only comparators and counters of the R/R + 1 type, that is:

$$\mathrm{R_C^B} = \mathrm{R_C^A} + 1 \implies \delta\mathrm{R_C^A} = \delta\mathrm{R_C^B} \tag{30}$$

it is possible to rewrite (27) as:

$$\mathrm{N_M} = \mathrm{R_C^A} \cdot \mathrm{I_A} + (\mathrm{R_C^A} + 1) \cdot \mathrm{I_B} \tag{31}$$

Therefore, taking into account (16), we can write

$$\mathrm{R_C^A} = \left\lfloor \frac{t_{\mathrm{ST}}}{t_{\mathrm{RTC}}} \right\rfloor \tag{32}$$

Thus, (29) becomes:

$$\begin{cases} \mathrm{R_C^A} \cdot \mathrm{I_A} + (\mathrm{R_C^A} + 1) \cdot \mathrm{I_B} = \mathrm{N_M} \\ \mathrm{I_A} + \mathrm{I_B} = \left\lfloor \dfrac{\mathrm{T_M}}{t_{\mathrm{ST}}} \right\rceil \\ \mathrm{R_C^B} = \mathrm{R_C^A} + 1 \\ \mathrm{R_C^A} = \left\lfloor \dfrac{t_{\mathrm{ST}}}{t_{\mathrm{RTC}}} \right\rfloor \end{cases} \tag{33}$$

By substituting and by using (26), we obtain:

$$\begin{cases} \mathrm{I_A} = \left\lfloor \dfrac{\mathrm{T_M}}{t_{\mathrm{ST}}} \right\rceil \left(\left\lfloor \dfrac{t_{\mathrm{ST}}}{t_{\mathrm{RTC}}} \right\rfloor + 1 \right) - \left\lfloor \dfrac{\mathrm{T_M}}{t_{\mathrm{RTC}}} \right\rceil \\ \mathrm{I_B} = \left\lfloor \dfrac{\mathrm{T_M}}{t_{\mathrm{RTC}}} \right\rceil - \left\lfloor \dfrac{\mathrm{T_M}}{t_{\mathrm{ST}}} \right\rceil \left\lfloor \dfrac{t_{\mathrm{ST}}}{t_{\mathrm{RTC}}} \right\rfloor \\ \mathrm{R_C^B} = \mathrm{R_C^A} + 1 \\ \mathrm{R_C^A} = \left\lfloor \dfrac{t_{\mathrm{ST}}}{t_{\mathrm{RTC}}} \right\rfloor \end{cases} \tag{34}$$

and, by substituting and simplifying:

$$\begin{cases} \mathrm{I_A} = \left\lfloor \dfrac{\mathrm{T_M}}{t_{\mathrm{ST}}} \right\rceil - \mathrm{I_B} \\ \mathrm{I_B} = \left\lfloor \dfrac{\mathrm{T_M}}{t_{\mathrm{RTC}}} \right\rceil - \left\lfloor \dfrac{\mathrm{T_M}}{t_{\mathrm{ST}}} \right\rceil \mathrm{R_C^A} \\ \mathrm{R_C^B} = \mathrm{R_C^A} + 1 \\ \mathrm{R_C^A} = \left\lfloor \dfrac{t_{\mathrm{ST}}}{t_{\mathrm{RTC}}} \right\rfloor \end{cases} \tag{35}$$

Solving the system (35) leads to a single solution

$$\mathbf{S} = \{\mathrm{I_A}, \mathrm{I_B}, \mathrm{R_C^A}, \mathrm{R_C^B}\} \tag{36}$$

that represents the initialization vector for the fractional timer.

C. RECEIVED MESSAGES QUEUE MANAGEMENT (RMQM) ROUTINE

Every $\mathrm{T_M}$, the master broadcasts to the slaves a synchronization packet containing its ST. Each slave analyzes and processes the received packet according to the algorithm shown in Fig. 3.

In Section II-D it was shown that when analyzing the case of a WBAN with broadcast timing message dissemination, the message delay reduces to the expression (8). Given its random nature, to reduce the reception time t_R at its minimum, the RMQM routine processes the synchronization packet with the highest priority with respect to every other possible type

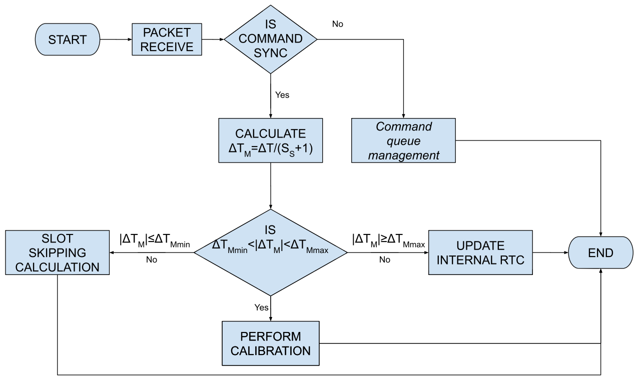
START
PACKET RECEIVE
IS COMMAND SYNC
No
Yes
CALCULATE $\Delta T_M = \Delta T/(S_S+1)$
Command queue management
SLOT SKIPPING CALCULATION
$|\Delta T_M| \leq \Delta T_{Mmin}$
No
IS $\Delta T_{Mmin} < |\Delta T_M| < \Delta T_{Mmax}$
$|\Delta T_M| \geq \Delta T_{Mmax}$
No
UPDATE INTERNAL RTC
END
Yes
PERFORM CALIBRATION

FIGURE 3. **RMQM routine flowchart.**

of message in the queue. Once the synchronization message is received, the slave j first evaluates the value:

$$\Delta\mathrm{T_M}(j) = \frac{\Delta\mathrm{T}(j)}{\mathrm{S_S}(j) + 1} \tag{37}$$

that represents the equivalent accumulated time difference between the master and the slave in a single slot $\mathrm{T_M}$. Therefore, the slave considers in the evaluation the number of skipped-slots $\mathrm{S_S}(j) = 0, 1, 2, .., k \in \mathbb{N}$ in which the radio section was turned off according to the slot-skipping routine. The absolute value of the obtained quantity $\Delta\mathrm{T_M}$ is compared with two reference values. The first, $\Delta\mathrm{T_{M,min}}$, represents the fixed lower bound for the synchronization error upon which a calibration is necessary and is determined by the required precision. In particular, the master and the slaves are considered to be synchronous when

$$|\Delta\mathrm{T_M}| \leq \Delta\mathrm{T_{M,min}} \tag{38}$$

and the slot skipping routine is executed.

The second value, $\Delta T_{\mathrm{M,max}}$, is set equal to $\mathrm{T_M}/2$. When the slave exceeds such value it simply updates its internal ST with the received value, since it is evident that the time difference is so high that the calibration process would be too expensive. In fact, this step is particularly useful in the early synchronization stage, when the values can be quite different.

In this sense, FLSA operates as an offset compensation algorithm. However, it will be clarified later that the algorithm actually operates also a skew compensation. This is achieved by changing the registers and counters values.

D. SLOT-SKIPPING ROUTINE

When the condition (38) occurs, it is possible to consider the slave to be synchronous to the master. Under this hypothesis, it is possible to save energy by switching off the radio section of the node, according to the slot skipping routine in Fig. 4. The routine operates in terms of synchronization slots. First, the number of slots in which the node can be considered to be synchronous to the master, $\mathrm{S_{sync}}(j)$, it is updated by adding the current slot to the previous value. The skipped slots (i.e. the ones in which the radio section is turned off)

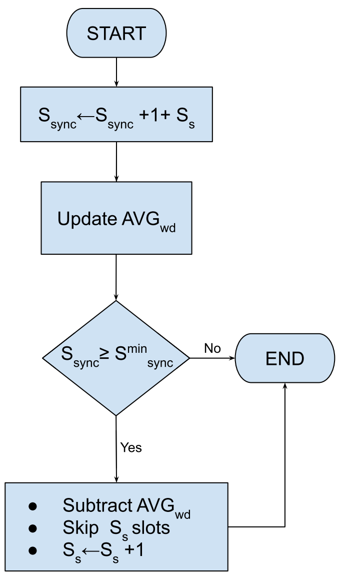

FIGURE 4. **Flowchart of the slot-skipping routine.**

are considered to be synchronous by hypothesis. To lighten the notation, in this paragraph we will consider each variable referred to the node j. The radio is switched off when the number of synchronous slots exceeds a predefined threshold called minimum slot skipped, S^{min}_{sync}. For the routine-call event k, the weighted mean value of the delays, $AVG_{wd}[k]$, is evaluated as

$$\mathrm{AVG}_{\mathrm{wd}}[k] = \frac{\mathrm{AVG}_{\mathrm{wd}}[k-1]S_{\mathrm{sync}}[k-1] + \Delta\mathrm{T}[k]}{\mathrm{S}_{\mathrm{sync}}[k]} \tag{39}$$

where

$$\mathrm{S}_{\mathrm{sync}}[k] = \mathrm{S}_{\mathrm{sync}}[k-1] + \mathrm{S}_{\mathrm{S}}[k] + 1 \tag{40}$$

For each iteration k the two values $\mathrm{AVG}_{\mathrm{wd}}[k]$ and $\mathrm{S}_{\mathrm{sync}}[k]$ are stored.

Because of the synchronization condition (38), the following inequality is always met:

$$|\mathrm{AVG}_{\mathrm{wd}}[k]| \leq \Delta\mathrm{T}_{\mathrm{M,min}} \tag{41}$$

During the slot skipping phase the radio is turned off, and at the end of each synchronization slot, the ST is adjusted taking into account the computed $\mathrm{AVG}_{\mathrm{wd}}[k]$, since it represents the average error for each synchronization slot, without calling the calibration routine. The computed $\mathrm{AVG}_{\mathrm{wd}}[k]$ is subtracted in each synchronization interval skipped and the skipped slot count is refreshed. The routine ends with the radio being switched on for a new synchronization packet, at the reception of which, the RMQM routine is executed.

It is possible to see how the skipped slot number linearly increases when the systems are synchronous. For example if we consider the node to be already synchronous, and the threshold $S^{min}_{sync} = 5$, the radio is switched on for 5 slots, off for 5 more, on for 1 more, off for 6 more, on for 1 more, off for 7 more, and so on. The more the system stays synchronous, the more evident the advantages are.

E. SLAVE TIMER CORRECTION ROUTINE

Let us suppose we are in the case that a correction is needed and the value $\Delta\mathrm{T}_{\mathrm{M}}(j)$ can be expressed equivalently in terms of number of ticks referred to the ticks that compose the time interval between two synchronizations as

$$\Delta\mathrm{N}_{\mathrm{M}} = \frac{\Delta\mathrm{T}_{\mathrm{M}}(j)}{t_{\mathrm{RTC}}(j)} \tag{42}$$

with $t_{\mathrm{RTC}}(j)$ being the RTC-time of node j. The calibration stage is governed by the set of equations:

$$\begin{cases} \mathrm{R}^{\mathrm{A}}_{\mathrm{C,new}}\,\mathrm{I}^{\mathrm{A}}_{\mathrm{new}} + \mathrm{R}^{\mathrm{B}}_{\mathrm{C,new}}\,\mathrm{I}^{\mathrm{B}}_{\mathrm{new}} = \mathrm{N}_{\mathrm{M}} + \Delta\mathrm{N}_{\mathrm{M}} \\ \delta R^{A}_{\mathrm{C}} = \delta R^{B}_{\mathrm{C}} \\ \mathrm{I}^{\mathrm{A}}_{\mathrm{new}} + \mathrm{I}^{\mathrm{B}}_{\mathrm{new}} = \left\lfloor \frac{\mathrm{T}_{\mathrm{M}}}{t_{\mathrm{ST}}} \right\rfloor \\ \{\mathrm{I}^{\mathrm{i}}_{\mathrm{new}}, \mathrm{R}^{\mathrm{i}}_{\mathrm{C,new}}\}_{i=\{\mathrm{A,B}\}} > 0 \end{cases} \tag{43}$$

with $\mathrm{R}^{\mathrm{i}}_{\mathrm{C,new}} = \mathrm{R}^{\mathrm{i}}_{\mathrm{C}} + \delta\mathrm{R}^{\mathrm{i}}_{\mathrm{C}}$ and $\mathrm{I}^{\mathrm{i}}_{\mathrm{new}} = \mathrm{I}_{\mathrm{i}} + \delta\mathrm{I}_{\mathrm{i}}$.

Considering that $\mathrm{I}_{\mathrm{A}} + \mathrm{I}_{\mathrm{B}} = \left\lfloor \frac{T_M}{t_{\mathrm{ST}}} \right\rfloor$, we can rewrite (43) as:

$$\begin{cases} \mathrm{R}^{\mathrm{A}}_{\mathrm{C,new}}\,\mathrm{I}^{\mathrm{A}}_{\mathrm{new}} + \mathrm{R}^{\mathrm{B}}_{\mathrm{C,new}}\,\mathrm{I}^{\mathrm{B}}_{\mathrm{new}} = \mathrm{N}_{\mathrm{M}} + \Delta\mathrm{N}_{\mathrm{M}} \\ \delta R^{A}_{\mathrm{C}} = \delta R^{B}_{\mathrm{C}} \\ \delta I_A = -\delta I_B \end{cases} \tag{44}$$

Doing the calculations, (44) becomes:

$$\begin{cases} \delta\mathrm{I}_{\mathrm{B}} = \Delta\mathrm{N}_{\mathrm{M}} - \delta\mathrm{R}^{\mathrm{B}}_{\mathrm{C}} \cdot \mathrm{N}_{\mathrm{M}} \\ \delta\mathrm{R}^{\mathrm{A}}_{\mathrm{C}} = \delta\mathrm{R}^{\mathrm{B}}_{\mathrm{C}} \\ \delta\mathrm{I}_{\mathrm{A}} = -\delta\mathrm{I}_{\mathrm{B}} \end{cases} \tag{45}$$

and, by substituting (26):

$$\begin{cases} \delta\mathrm{I}_{\mathrm{B}} = \Delta\mathrm{N}_{\mathrm{M}} - \delta\mathrm{R}^{\mathrm{B}}_{\mathrm{C}} \cdot \left\lfloor \frac{\mathrm{T}_{\mathrm{M}}}{t_{\mathrm{RTC}}} \right\rfloor \\ \delta\mathrm{R}^{\mathrm{A}}_{\mathrm{C}} = \delta\mathrm{R}^{\mathrm{B}}_{\mathrm{C}} \\ \delta\mathrm{I}_{\mathrm{A}} = -\delta\mathrm{I}_{\mathrm{B}} \end{cases} \tag{46}$$

The system has ∞^1 solutions. To find one of the possible solutions, an iterative procedure can be used, starting by imposing:

$$\delta\mathrm{R}^{\mathrm{A}}_{\mathrm{C}} = \delta\mathrm{R}^{\mathrm{B}}_{\mathrm{C}} = 0 \tag{47}$$

and verifying that the boundary conditions are satisfied. Therefore,

$$\begin{cases} \delta\mathrm{I}_{\mathrm{B}} = \Delta\mathrm{N}_{\mathrm{M}} \\ \delta\mathrm{R}^{\mathrm{A}}_{\mathrm{C}} = \delta R^{\mathrm{B}}_{\mathrm{C}} = 0 \\ \delta\mathrm{I}_{\mathrm{A}} = -\Delta\mathrm{N}_{\mathrm{M}} \end{cases} \tag{48}$$

The new parameter to be set is available only when the system has a solution,i.e. $|\Delta\mathrm{N}_{\mathrm{M}}| \leq \delta\mathrm{I}_{\mathrm{i}}$, $i = \{A, B\}$. In particular, the subroutine checks if a simple variation of the I_{A} and I_{B} counters is sufficient. Otherwise, it varies the $\mathrm{R}^{\mathrm{A}}_{\mathrm{C}}$ and $\mathrm{R}^{\mathrm{B}}_{\mathrm{C}}$ dividers values according to the increment or decrement required by ST. Therefore, depending on the sign of $\Delta\mathrm{N}_{\mathrm{M}}$, the various possibilities reported in the flow-chart in Fig. 5 may occur. To set the new counter values, the initialization

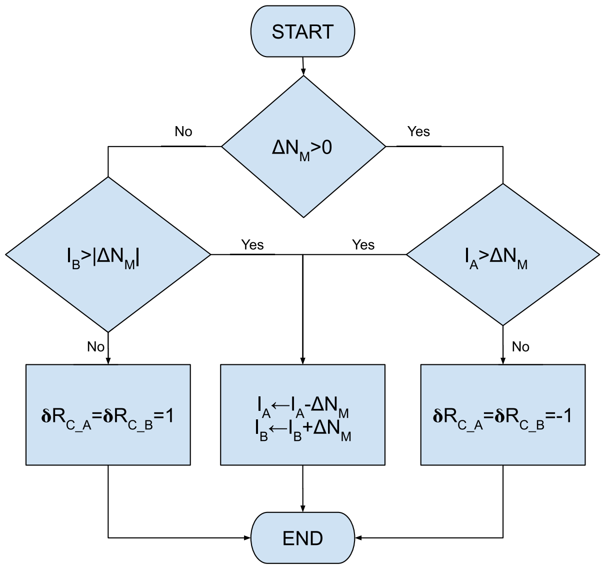
START
$\Delta N_M>0$
No
Yes
$I_B>|\Delta N_M|$
Yes
Yes
$I_A>\Delta N_M$
No
No
$\delta R_{C\_A}=\delta R_{C\_B}=1$
$I_A \leftarrow I_A-\Delta N_M$
$I_B \leftarrow I_B+\Delta N_M$
$\delta R_{C\_A}=\delta R_{C\_B}=-1$
END

FIGURE 5. Timer correction routine flowchart.

step described in (35) is then modified and executed as follows:

$$\begin{cases} I_A = \left\lfloor \dfrac{T_M}{t_{ST}} \right\rceil - I_B \\ I_B = N_M + \Delta N_M - \left\lfloor \dfrac{T_M}{t_{ST}} \right\rceil R_C^{A,new} \end{cases} \tag{49}$$

It is very important to highlight that the method detects the t_{ST} minimum deviation while the correction precision is equal to t_{RTC}, that is, the minimum allowed value determined by the specific hardware implementation. It is also evident that a correction on both registers and counters is equivalent to a skew correction. Referring to the model described in Section II-B, changing those values is equivalent to set the new value α_i referring to α_j. Note that, as already mentioned, the timer correction routine is not called-back when the slot-skipping routine is executed.

VI. VALIDATION OF THE ALGORITHM

The algorithm validation has been performed using the IMUboard-HW-1-0-0 unit described in [14]. The unit was designed and assembled integrating 9 DoF MPU-9250 nine-axis MEMS MotionTracking^{TM}device, produced by TDK-InveSense. This component is a System in Package (SiP) that combines two chips: the MPU-6500, which contains a 3-axis gyroscope, a 3-axis accelerometer, and an onboard processor, and the AK8963, a 3-axis digital magnetometer. The chips are mutually connected using the AUX-I2C of the MPU-6500 [27]. The range of angular velocities of the SiP covers the span from ±250 [deg/s] up to ±2000 [deg/s]; that of the acceleration covers the ±2[g] to ±16[g] interval; the magnetometer measures up to ±4800[μT]; the sampling rate arrives to 8 kHz and the communication clock frequency is up to 20 MHz for the SPI reading protocol. The core of the IMUboard-HW-1-0-0 unit is the Nordic Semiconductor SoC nRF52832 that integrates a 64 MHz ARM®Cortex^{TM}-M4 CPU with 512 kB Flash memory and 64 kB RAM. This SoC manages a multiprotocol Bluetooth 5, ANT/ANT+ and proprietary ISM 2.4 GHz. The unit allocates a micro-SD card for data logging, controlling it in SPI bus mode using the FAT32 as file system to allow the direct download of the data on a PC. A couple of LEDs (green and red) represent the user interface along with two push-button switches for direct interaction and one main power switch. The unit is powered either by a CR2450 coin cell battery. The IMUboard-HW-1-0-0 unit is depicted in Figure 6. We experimentally proved both the low-power and the high accuracy of the proposed technique using a long time observation interval. We ran two types of experiments. In the first type of experiment, we considered a master that periodically sent its ST to a slave. In turn, the slave outputs the synchronization results to a PC connected via a serial port. For the second experiment, to consider different initial conditions and their effects on the overall synchronization accuracy and speed, we used three different slaves, each of them connected to the PC via a serial port.

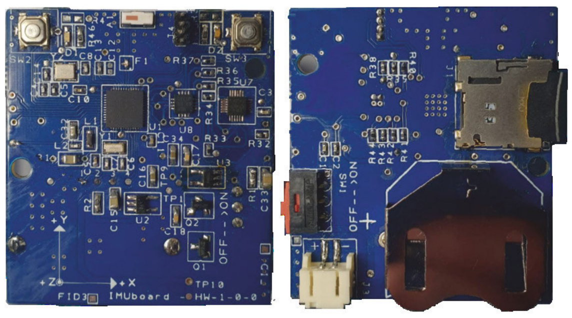

FIGURE 6. Top and bottom views of IMUboard-HW-1-0-0 unit.

For both experiments, a seven days measurement campaign has been performed and during this long observation period, the slaves sent on the serial connection all the information regarding the algorithm implementation and, in particular, the number of skipped slots (when perfect synchronization with the master was achieved) and the register values regarding the fractional timer implementation. In Table 3 it is possible to find the summary of the boards and algorithm parameters that have been used to run the experiments (f_{IMU} is the sampling frequency of the IMU sensors).

TABLE 3. Parameters employed for the experiments.

T_M	20 s
t_{RTC}	30.517 µs
t_{ST}	100 µs
ΔT_{min}	500 µs
ΔT_{max}	5 s
S_{sync}^{min}	5
f_{IMU}	100 Hz

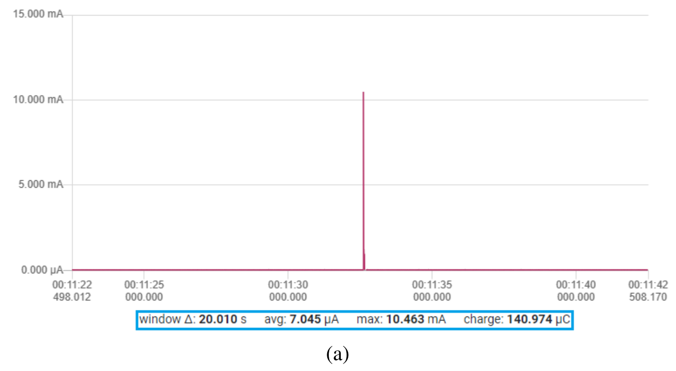

FIGURE 7. Current measurements performed with the PPK: (a) entire $T_M = 20$ s period; (b) a 34.783μs duration window of the synchronization phase.

A. POWER CONSUMPTION

The power consumption analysis has been performed using the Power Profiler Kit (PPK) [28], which is a low cost, highly accurate power measurement tool from Nordic Semiconductor. The PPK measures the current drained by the custom board. All measurements refer to the same power supply voltage of 3V DC. The current drained by the unit has been measured while receiving the packet, during the processing phase and with the radio switched-off (stand-by phase). These scenarios correspond to the main states of the algorithm. Fig. 7 and Fig. 8 show the corresponding measurements. Fig. 7a shows the slave current monitored over a $T_M = 20$ s interval, with a single reception event, while Fig. 7b zooms over the synchronization packet processing phase.

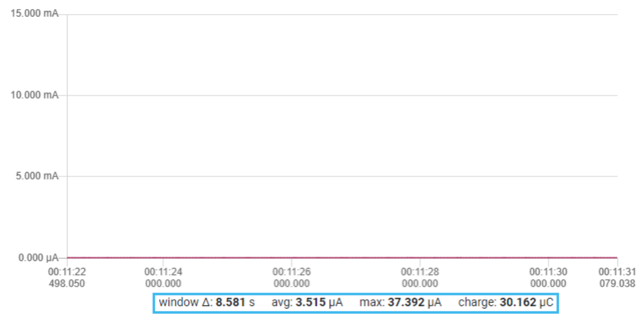

FIGURE 8. Current measurement during the stand-by phase.

The measured values averaged over the whole $T_M = 20$ s synchronization interval give a 7.045 μA mean drained current. By averaging the drained current during the synchronization phase (whose duration is 34.783 ms), when the radio is switched on a 2.1 mA. Figure 8 shows that with the radio switched-off between two packet receptions the average current is 3.515 μA. These measured values and the data collected during the previous measurements campaign were processed using a MATLAB script to evaluate the algorithm performances.

The maximum number of skipped slots is 53, as it will be shown in next section. Fig. 9 reports the instantaneous power consumption evaluation of the slave when the algorithm is not

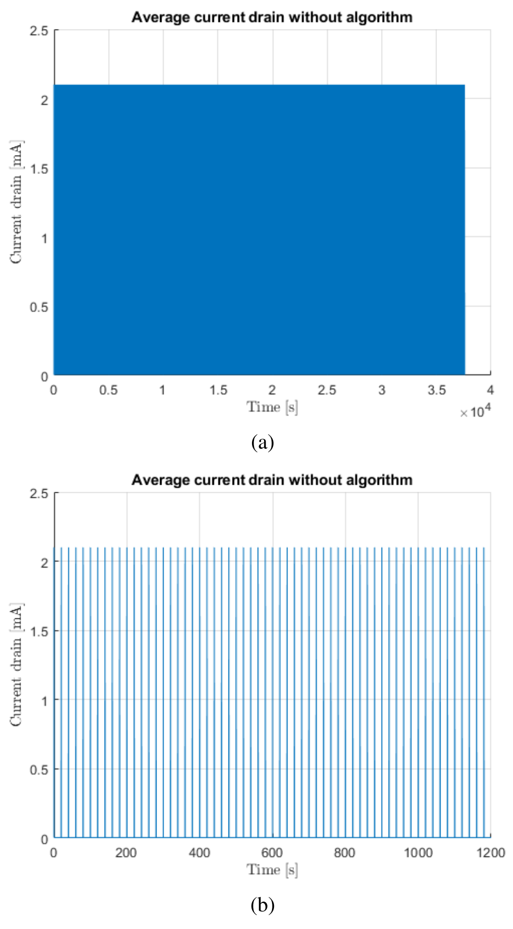

FIGURE 9. Simulation of drained current without algorithm: global view (a) and zoom (b).

executed. In particular, Fig. 9a shows the global view, while Fig. 9b shows the drained current at 20 s intervals. Fig. 10 reports the current drained during the same time interval of Fig. 9a but with the algorithm in use. For each iteration the radio activation is delayed by a quantity always greater than the skipped intervals as explained in the Section V-D. Reduction in radio resource access brings to a significant power consumption reduction.

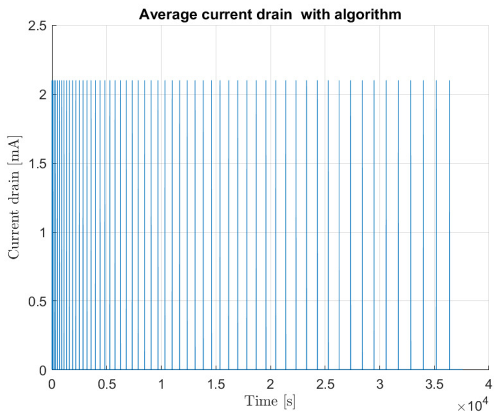

FIGURE 10. Simulation of drained current with algorithm.

Fig. 11 quantifies the power saving, reporting the average drained current as a function of the skipped slots. The sharp current reduction results in a graph that has an hyperbolic trend.

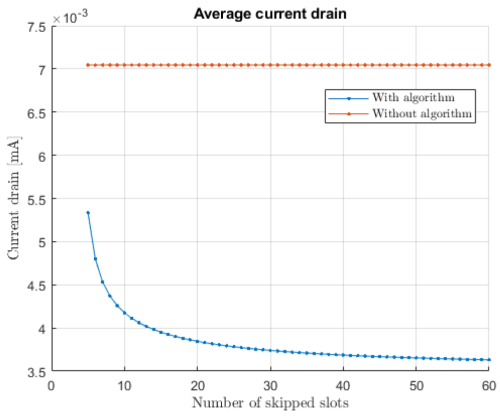

FIGURE 11. Comparison of the average current drained when the algorithm is adopted with respect to periodic synchronization packet transmission.

What previously discussed is even more evident in Fig. 12 that reports the algorithm efficiency as a function of the skipped slots in the form of percentage power reduction with respect to the case in which the algorithm is not used. The algorithm effects are immediately apparent and there is a 50% saving as soon as the algorithm starts up. The saving rapidly increases as the number of skipped slots increase, and asymptotically tends to be constant when this number further increases. Fig. 11 e Fig.12 report the results after the first 5 skipped slots. The algorithm settings are listed in Table 3.

B. SYNCHRONIZATION PERFORMANCE

To validate the synchronization performance in terms of accuracy and speed of the method, we used 3 slaves, each of them starting by different initial conditions. Every slave and the master were hosted on nominally identical boards. Table 4 reports the ideal initial values (IV) computed by the master

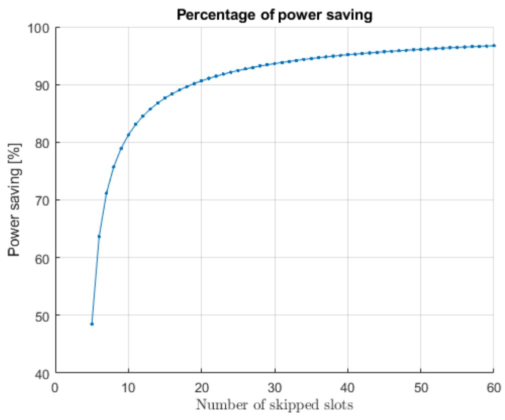

FIGURE 12. Percentage of power savings as function of the skipped slots.

TABLE 4. Internal register and counter values (CVs) for the three different slaves.

Register	IV	CV Slave#1	CV Slave#2	CV Slave#3
R_C^A	3	3	3	3
R_C^B	4	4	4	4
I_A	144263	144503	144602	144018
I_B	55737	55497	55398	55982

and the slaves. The master took into account these values all along the experiment. Table 4 also shows the values computed by the slaves to synchronize themselves with the master.

The first consideration is that nominally identical units exhibit temporal mismatches even when operating in the same conditions [29]. Moreover, it is worth to underline that only two synchronization intervals were needed to obtain the calibrated values, demonstrating that the proposed algorithm is quite effective, as the synchronization to the master timer is achieved with only two synchronization messages. In particular, in this phase, the first message allowed for a first (coarse) synchronization, while the second one brought to a finer synchronization.

Algorithm performance and precision are a function of the adopted parameters. Let us refer to the values listed in Table 3. For each of the three slaves, we extracted the maximum, minimum and average number of skipped slots listed in Table 5, during the 7-days measurement campaign. These values were taken as a reference to derive the algorithm precision. In fact, if we consider, for instance, Slave #1, the maximum number of skipped slots before requiring a new calibration is equal to 53. Thus, for 1474 consecutive intervals (i.e., 29480s) the time mismatch was lower than $\Delta T_{min} = 500\,\mu s$, according to what we discussed in Section V. The worst case was reported with Slave #1 that during one of the sessions skipped only 12 slots. However, that means that in the worst case, the minimum time interval in which the whole network was synchronized over a 7-days long experiment was 1600 s. The power drained during this measurement campaign along with the number of skipped slots is reported in Table 5. It is evident

TABLE 5. Experimental results, with S_s denoting the number of skipped slots, I the current and P the power saving.

	Slave #1	Slave #2	Slave #3
max S_s	53	49	51
min S_s	12	18	14
avg S_s	39	42	37
max I [μA]	4.06	3.881	3.982
min I [μA]	3.646	3.656	3.651
avg I [μA]	3.690	3.678	3.699
min P [%]	96.405	96.124	96.269
max P [%]	85.000	89.944	87.156
avg P [%]	95.179	95.508	94.931

that the minimum percentage power saving obtained on the whole network amounts to 85%, and it corresponds to the Slave #1 worst synchronization case. Yet, the data show that the maximum power saving slightly varies around the 96% for each slave of the network.

VII. CONCLUSION AND FUTURE WORK

The paper introduced the FLSA which allows performing the time synchronization between the nodes of a WBAN. The algorithm is based on the master-slave paradigm and it ensures an extremely low-power consumption without loss of accuracy, even on a long time period, as proven by the reported measurement campaigns.

One of the advantages of the proposed algorithm is that it can be easily implemented in any low-cost and low-power system thanks to its very low computational complexity. As an example, the paper proposes an implementation on a SoC integrating a low-power ARM®Cortex$^{\text{TM}}$-M4. It has been experimentally demonstrated that at start-up, even if nominally identical boards start from different conditions, the algorithm quickly reaches synchronization and maintains it for long periods of time, being able to exploit the maximum precision made available through the hardware.

Future work can be done evaluating more complex schemes of fractional implementation such as tri-modulus and in general multi-modulus, evaluating the impact of the possible evolutions in terms of accuracy and computational complexity. The use of a multi-modulus systems would also make the randomization of modulus control more effective.

Another topic subject to future work will be the multi-hop expansion of the proposed approach.

REFERENCES

[1] H. M. Ammari, *The Art of Wireless Sensor Networks: Fundamentals*, vol. 1. Berlin, Germany: Springer, 2014.

[2] G. Coviello, G. Avitabile, A. Florio, and C. Talarico, "A study on IMU sampling rate mismatch for a wireless synchronized platform," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2020, pp. 229–232.

[3] G. Mehmood, M. Z. Khan, S. Abbas, M. Faisal, and H. U. Rahman, "An energy-efficient and cooperative fault-tolerant communication approach for wireless body area network," *IEEE Access*, vol. 8, pp. 69134–69147, 2020.

[4] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour, "Wireless body area networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1658–1686, 3rd Quart., 2014.

[5] F. Gong and M. L. Sichitiu, "On the accuracy of pairwise time synchronization," *IEEE Trans. Wireless Commun.*, vol. 16, no. 4, pp. 2664–2677, Apr. 2017.

[6] *Network Time Protocol (Version 3) Specification, Implementation and Analysis*, document RFC 1305, Mar. 1992. [Online]. Available: https://rfc-editor.org/rfc/rfc1305.txt

[7] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588-2019 (Revision of IEEE Standard 1588-2008), 2020, pp. 1–499.

[8] T. Yang, Y. Niu, and J. Yu, "Clock synchronization in wireless sensor networks based on Bayesian estimation," *IEEE Access*, vol. 8, pp. 69683–69694, 2020.

[9] H. Zhu, K. Liu, Y. Yan, H. Zhang, and T. Huang, "Measures to improve the accuracy and reliability of clock synchronization in time-sensitive networking," *IEEE Access*, vol. 8, pp. 192368–192378, 2020.

[10] X. Chen, D. Li, S. Wang, H. Tang, and C. Liu, "Frequency-tracking clock servo for time synchronization in networked motion control systems," *IEEE Access*, vol. 5, pp. 11606–11614, 2017.

[11] S. Huang, N. Zheng, Y. Wu, and M. Xu, "Resilient consensus-based time synchronization in asynchronous sensor networks," *IEEE Access*, vol. 7, pp. 115650–115661, 2019.

[12] *ISO/IEC/IEEE International Standard—Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 15–6: Wireless Body Area Network*, Standard ISO/IEC/IEEE 8802-15-6:2017(E), 2018, pp. 1–274.

[13] Z. Yang, L. He, L. Cai, and J. Pan, "Temperature-assisted clock synchronization and self-calibration for sensor networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 6, pp. 3419–3429, Jun. 2014.

[14] G. Coviello, G. Avitabile, and A. Florio, "A synchronized multi-unit wireless platform for long-term activity monitoring," *Electronics*, vol. 9, no. 7, p. 1118, Jul. 2020.

[15] G. Coviello and G. Avitabile, "Multiple synchronized inertial measurement unit sensor boards platform for activity monitoring," *IEEE Sensors J.*, vol. 20, no. 15, pp. 8771–8777, Aug. 2020.

[16] K. S. Yıldırım and Ö. Gürcan, "Efficient time synchronization in a wireless sensor network by adaptive value tracking," *IEEE Trans. Wireless Commun.*, vol. 13, no. 7, pp. 3650–3664, Jul. 2014.

[17] A. Al-Shaikhi and A. Masoud, "Efficient, single hop time synchronization protocol for randomly connected WSNs," *IEEE Wireless Commun. Lett.*, vol. 6, no. 2, pp. 170–173, Apr. 2017.

[18] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 147–163, Dec. 2002.

[19] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 39–49.

[20] L. He and G.-S. Kuo, "A novel time synchronization scheme in wireless sensor networks," in *Proc. IEEE 63rd Veh. Technol. Conf.*, vol. 2, May 2006, pp. 568–572.

[21] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. 1st Int. Conf. Embedded Netw. Sensor Syst.*, 2003, pp. 138–149.

[22] H. Wang, H. Zeng, and P. Wang, "Linear estimation of clock frequency offset for time synchronization based on overhearing in wireless sensor networks," *IEEE Commun. Lett.*, vol. 20, no. 2, pp. 288–291, Feb. 2016.

[23] K. Noh, E. Serpedin, and K. Qaraqe, "A new approach for time synchronization in wireless sensor networks: Pairwise broadcast synchronization," *IEEE Trans. Wireless Commun.*, vol. 7, no. 9, pp. 3318–3322, Sep. 2008.

[24] H. Kawagoe and M. Sugano, "Implementation of time synchronization for energy harvesting wireless sensor network," in *Proc. VI Int. Conf. Netw., Commun. Comput. (ICNCC)*, New York, NY, USA, 2017, pp. 175–178.

[25] *nRF52832 Product Specification V1.4*. Accessed: Jan. 7, 2021. [Online]. Available: https://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.4.pdf

[26] B. Miller and R. Conley, "A multiple modulator fractional divider," *IEEE Trans. Instrum. Meas.*, vol. 40, no. 3, pp. 578–583, Jun. 1991.

[27] *MPU9250 Datasheet*. Accessed: Jan. 7, 2021. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf

[28] *Power Profiler Kit*. Accessed: Jan. 7, 2021. [Online]. Available: https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/PowerProfilerKit-product-brief.pdf

[29] G. Coviello, G. Avitabile, and A. Florio, "The importance of data synchronization in multiboard acquisition systems," in *Proc. IEEE 20th Medit. Electrotech. Conf. (MELECON)*, Jun. 2020, pp. 293–297.

GIUSEPPE COVIELLO (Member, IEEE) was born in Bari, Italy, in 1981. He received the Laurea degree in electronic engineering and the Ph.D. degree in electric and information engineering from the Polytechnic University of Bari, Italy, in 2006 and 2015, respectively. He is currently an Assistant Professor with the Department of Electrical and Information Engineering, Polytechnic University of Bari. His research interests include RF and mixed-signal circuits and embedded systems for healthcare.

GIANFRANCO AVITABILE (Senior Member, IEEE) was born in Livorno, Italy, in 1958. He received the B.S. and M.S. degrees in electronics from the University of Florence, Italy, in 1982. He joined the Department of Electronics, University of Florence, as an Assistant Professor. In 1998, he joined the Department of Electronics, Polytechnic University of Bari, as an Associate Professor, where he currently works. His research interests include high-frequency circuits and systems for both civil and military applications, and high-performance analog integrated circuits for telecommunication applications.

ANTONELLO FLORIO (Student Member, IEEE) was born in Bari, Italy, in 1995. He received the B.Sc. degree (Hons.) in computer and automation engineering and the M.Sc. degree (Hons.) in telecommunications engineering from the Polytechnic University of Bari, Italy, and the M.Sc. degree in computer science with specialization in ubiquitous networking and computing from the University of Nice Sophia Antipolis, France. He is currently pursuing the Ph.D. degree in electronics with the Polytechnic University of Bari. His research interests include phased arrays, theories and techniques for localization, and green wireless sensor networks.

CLAUDIO TALARICO (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from the University of Genoa, Italy, and the Ph.D. degree in electrical engineering from the University of Hawaii. He is currently a Professor of electrical and computer engineering with Gonzaga University. Before joining Gonzaga University, he was with Eastern Washington University and The University of Arizona, and industry, where he held both engineering and management positions at Siemens Semiconductors, IKOS Systems, and Marconi Communications. His research interests include digital and mixed analog/digital integrated circuits and systems, computer-aided design methodologies, and design and analysis of systems-on-chip.

JANET M. WANG-ROVEDA received the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California at Berkeley, in 1998 and 2000, respectively. She is currently an Associate Professor with the Department of Electrical and Computer Engineering, The University of Arizona, Tucson. She was a recipient of NSF Career Award and the Presidential Early Achievement Award for Science and Engineering at White House, in 2005 and 2006, respectively, the 2008 R. Newton Graduate Research Award from EDA Community, and the 2007 U.S.S. University of Arizona Outstanding Achievement Award. Her primary research interests include robust VLSI circuit design, biomedical instrument design, smart grid, VLSI circuit modeling/design and analysis, and low-power multi-core system design.