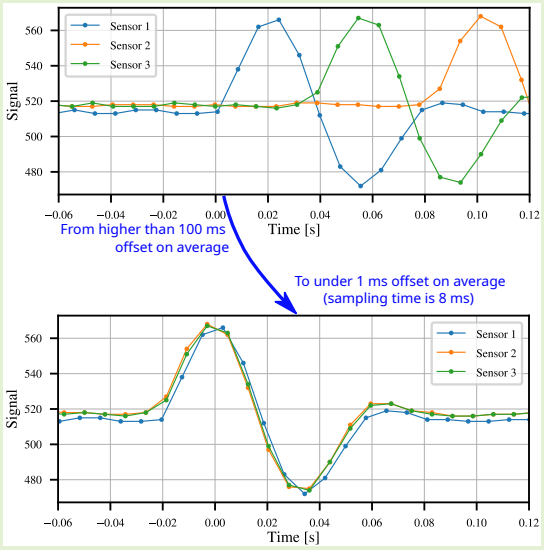
# Offline synchronization of signals from multiple wireless sensors

Matjaž Depolli, Nina Verdel, and Gregor Kosec

***Abstract*— This study addresses the critical challenges of time synchronization in wearable sensor networks, focusing on electrocardiogram (ECG) and inertial measurement unit (IMU) monitoring applications. In the era of continuous physiological and biomechanical monitoring, accurate time synchronization of sensor data is critical. Our research investigates the effectiveness of offline synchronization method chosen for its flexibility and precision in addressing time-related anomalies in environments where real-time processing of the gathered data is not required. The synchronization method works independently for each node without exchanging time-sync packets among nodes, only among nodes and a central device. We present a synchronization approach that has been designed to deal with variable sampling frequency, random transmission delay and packet loss. We demonstrate the efficiency of the approach with two different example applications: long-term ECG monitoring and short-term IMU-based gait analysis. The example applications use different strategies for storing the sampled data and for exchanging time-sync packets. Our results show that the proposed synchronization method is robust and accurate. We identify the limit for accuracy to be in the communication software of the master device and sensor nodes. This study contributes to the field of wearable sensor networks by presenting a comprehensive synchronization method.**




***Index Terms*— sensors, wireless, synchronization, Bluetooth, resampling, time alignment**

## I. Introduction

WEARABLE technology has revolutionized the field of continuous physiological and biomechanical monitoring in both clinical and sports settings [1], [2]. This enables long-term tracking of an individual's health and fitness parameters, providing valuable insights for the early detection and prevention of potential health issues. The advancements in telecommunications have further enhanced this capability by allowing wireless data transmission from miniature sensor devices to personal terminals like smartphones and tablets, enabling remote patient monitoring, medical support, remote sports coaching, and virtual coaching. These technological innovations have the potential to greatly transform wellness approaches while making a significant impact on both the healthcare and sports industries. [3]

Non-communicable diseases (NCD) contribute to a significant number of global deaths, with cardiovascular disease being the leading cause [4]. The prevalence of NCDs is higher in low- and middle-income countries, highlighting the importance of effective prevention and management strategies [5]. Close monitoring and tracking of NCD occurrence and progression are essential for developing effective interventions and treatment approaches. Cardiovascular diseases encompass various conditions, including cardiac arrhythmias such as ventricular tachycardia or ventricular fibrillation, which contribute to around 8% of cases [6]. The use of wearable continuous ECG monitoring has proven effective in the early detection and prevention of death caused by atrial fibrillation [7]. Since the 1960s, long-term ECG monitoring using Holter monitors has been commonly used in medical practice. In recent years, there has been significant progress in the development of wearable ECG monitoring devices. These devices have evolved from large wired systems to small wire-free wearables that are more convenient for patients and cause minimal disruption to daily activities. [8] This advancement aims to overcome the limitations of conventional Holter monitors and provide a more comfortable experience for users. One recent development in this area is the PCARD mHealth platform, which introduces a novel methodology for synthesizing 12-lead ECG measurements using three concurrent differential ECG measurements [9]. These readings can be obtained from three separate

This work was supported part by the Slovenian Research and Innovation Agency under grant agreement No. J5-3115, and research core funding No. P2-0095." Corresponding author: N. Verdel.

N. Verdel is with the Faculty of Sport, University of Ljubljana, 1000 Ljubljana, Slovenia and with the Department of Communication Systems, Jozef Stefan Institute, Ljubljana, Slovenia (email: nina.verdel@fsp.uni-lj.si).

M.Depolli and G. Kosec are with the Department of Communication Systems, Jozef Stefan Institute, Ljubljana, Slovenia (email: matjaz.depolli@ijs.si, gregor.kosec@ijs.si)

wearable sensor devices that are wirelessly connected to a single smartphone. This capability of wearable ECG sensors holds significant potential for expanding the availability of 12-channel ECG measurement outside of healthcare facilities.

Moreover, engaging in regular physical activity is essential for maintaining good health and a high quality of life while also preventing the onset of non-communicable diseases. Running, in particular, stands out as an efficient and cost-effective form of exercise. However, improper execution can result in injuries that negatively impact individual's overall well-being and lead to increased healthcare expenses and demands on the social support system [10], [11]. Coaching based on gait analysis during running, both indoors and outdoors, has shown promising potential in reducing the risk of injury.

Traditionally, gait analysis is conducted in a laboratory settings using gold-standard equipment such as motion capture system, force plates, and instrumented treadmills [12]. However, a subject's running cycle may differ from their natural stride under controlled laboratory conditions [13]. Recent advancements in wearable sensors have made it possible to assess gait more broadly in a natural, out-of-lab environment at a relatively low cost compared to the standard measurement methods. To ensure that athletes benefit from wearable sensors, the sensors must be wireless, lightweight, and small in size to prevent discomfort during running. Recently, one such wearable device aimed at preventing running injuries has been developed [14]. This device includes three IMU sensors strategically positioned on the foot (two senors) and the pelvic area.

Both wearable devices mentioned earlier require multiple sensor nodes that communicate with a smartphone for signal collection and processing. To ensure valuable information is obtained from all the nodes, signals from each node must be time synchronized [15]. While simple sensor nodes rely on a quartz oscillator for internal time representation, with an estimated accuracy of under 100 ppm [16], the smartphone's clock can be more accurate, depending on quartz oscillators with active compensation from time sources available through the Internet. When working as a system, the unaltered quartz accuracy of the nodes leads to significant time drift during long measurements; for instance, the discrepancy between the smartphone and the sensor may be on the order of seconds per day. Although this level of time drift can be acceptable for long-term measurements on a single sensor, it does not meet the requirements for taking multi-sensor measurements that need to be synchronized between each other. Therefore, a dynamic and adaptive strategy must be implemented to synchronize the clocks of the sensor nodes with that of the smartphone clock to allow for precise data synchronization.

There are two main methods for time synchronization [17]: online and offline time synchronization. In online synchronization, the involved nodes actively synchronize their clocks during measurement [18], while in offline synchronization, the involved nodes run on unsynchronized clocks and synchronization occurs after the data collection [19]. Offline synchronization has several advantages: it saves energy, which is critical for battery-powered sensor nodes; the sensors do not need to run complex synchronization algorithms in real time, which saves computational resources; and the entire data set is available to the synchronization algorithm during post-processing, potentially leading to improved accuracy. However, the drawback is that the measurements are only accessible after the post-processing, which cannot happen in real-time. Nevertheless, there are many different sports and medical applications that do not require access to real-time measurements and can benefit from the aforementioned advantages. In medicine, such applications can include monitoring movement (kinematic data) in patients with Parkinson's disease [20], [21], in neurologically impaired individuals [22], during lower rehabilitation [23], ECG monitoring, and more. Additionally, in sports, lower limb kinematics can be measured offline, allowing fir the determination of asymmetries related to performance and increased injury risk [24]–[27] can be determined afterwards. Therefore, in the presented study, we focus on applications that do not require the access to measurements in real-time, and propose a new approach for offline sensor synchronization. This proposed approach is based on previous work done on the same PCARD mHealth platform [28], [29], but it has been extended and generalized in this study to accommodate different platforms as well. However, in the discussion, our method will be compared to an offline approach published by [19], and an offline real-time synchronization approach used in PCARD mHealth platform.

The proposed methodology requires no trigger signal for the sensor nodes and no inter-nodal communication of time-sync packets. It works by converting sensor time to reference time for each involved sensor node separately, which makes it scale well with the number of sensors. The only requirements for sensors are that they keep their own local time and that they frequently exchange time-sync packets with a single master device.

## II. Problem Statement

Keeping accurate time, which is necessary for synchronizing events measured on different devices, is not trivial. Suppose a theoretical case in which each of the sensor nodes samples a signal at an accurate predetermined sampling frequency, packs several samples into communication packets, and transmits these packets at a constant rate over a perfect communication channel with no delay. Under these ideal conditions, the sampling time of each sample would be perfectly determined on the receiving end, making the synchronisation of multiple parallel measurements a trivial task. The actual situation differs from the ideal; wireless sensors rely on wireless communication, which introduces uncertainty in packet transmission. This type of communication is susceptible to packet loss, and transmission delays. The latter are seemingly random and occur due to the implementation of communication protocols, that sacrifices the user's ability to control some aspects of the communication, such as the exact time of packet transmission, for the overall ease of use.

Sensor node internal clocks depend on quartz oscillators. Although these oscillators are reasonably accurate, they are subject to slight variations in frequency due to changes in temperature, aging, or manufacturing disparities. Over time,

these small discrepancies can accumulate, leading to significant time dependent drift on a single node and discrepancies in the time reported by different nodes.

Both, the uncertainty related to the communication channel and variation in the accuracy of all involved clocks, result in the data packets being received at unpredictable times even if they were sent at regular and predetermined intervals according to the transmitting device. The above mentioned properties of sensor nodes creates the following challenges of keeping the devices synchronized:

- variable clock frequency,
- random delay,
- packet loss.

Random delays are shown in Figure 1 as the differences between the ideal case and reality, and packet losses are marked with grey crosses.

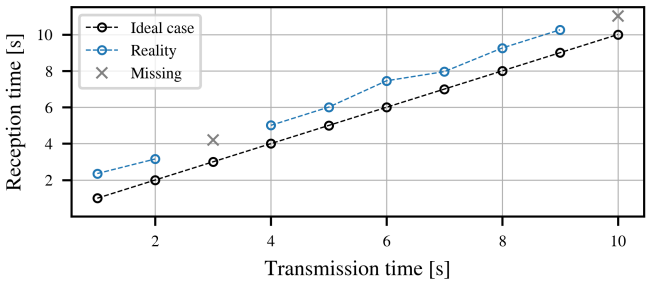



Fig. 1. An illustration of the timestamps sequences. Black circles – idealistic case with no transmission delays, cyan circles – realistic case where received timestamps are delayed, gray crosses – missing packets.

## A. Tracking time

Time synchronization between the sensor nodes and the master device is essential. Without synchronized time-stamps, data from different sensor nodes might not be comparable or might provide misleading information regarding the timing and sequence of the monitored events. While time synchronization can be achieved through real-time clock synchronization, this paper does not address that option. Instead, the proposed method relies on collecting time-related data for offline clock synchronization after the measurements have been completed. In the offline synchronization approach, the relation between the clocks or the function to translate time measured by one clock into the time measured by another is approximated from the available data. The minimal data required to support synchronization of two devices is a list of tuples comprising associated timestamps taken by the two involved clocks, Fig. 2:

- Timestamp one: This is recorded by device one, using device's local clock, just prior to making a transmission to device two. This timestamp is transmitted to device two either on its own or along with measurement sample that corresponds to the timestamp.
- Timestamp two: Upon receipt of timestamp one, device two takes its own timestamp immediately and records both timestamps.

These tuples provide critical data points. By analyzing the differences between timestamp one and timestamp two across multiple tuples, it's possible to determine the degree of clock drift and offset between the two clocks.

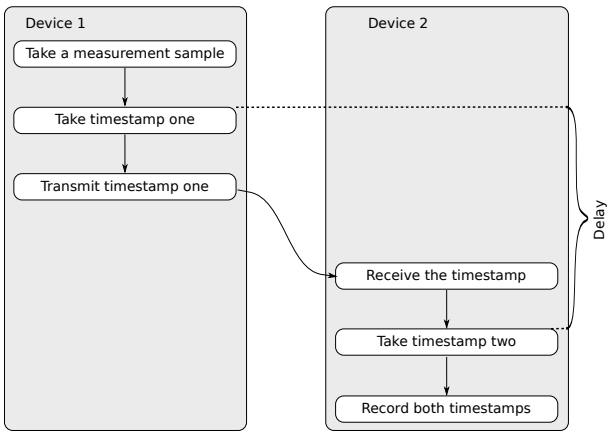



Fig. 2. A step of linear regression performed on a real measurement. Range of the displayed time is manually selected

Note that in the presented theory, either device can serve as a master device or as a sensor node.

## B. Signal sampling

Consider an idealistic sequence of timestamp tuples, where there are no delays between the recorded timestamps of each tuple and the samples are equidistant. In this idealistic case, the sampling frequency, denoted as $f_s$, is defined as follows:

$$f_s = \frac{1}{t_s}, \tag{1}$$

where $t_s$ represents the sampling period also called the sampling time, i.e., the time between two samples observed by the master device. Consequently, for the $i$-th sample in the sequence, the reference time $t(i)$ at which it was taken can be expressed as:

$$t(i) = \frac{i}{f_s} + t_{start}, \tag{2}$$

where $t_{start}$ is the measurement start time. However, sampling is based on the local clock of the sensor with a given resolution. The resolution of the sensor clock is called a tick, and the clock is usually simplified to the point it comprises only a tick counter. To implement sampling with frequency $f_s$, the length of a tick $t_{tick}$ is known in advance and the sampling time is converted into a tick count $n_s$:

$$n_s = \frac{t_s}{t_{tick}}. \tag{3}$$

Then the $i^{\text{th}}$ sample time can be expressed in sensor time $n(i)$ as a function of variables available to the sensor:

$$n(i) = n_s i + n_{start}, \tag{4}$$

where $n_{start}$ is the measurement start time expressed in sensor clock ticks. Finally Equations 2 and 4 can be combined to

express the reference time as a linear function of sensor time:

$$t(n) = t_{start} + \frac{n - n_{start}}{n_s f_s}, \tag{5}$$

where $n$ now acts as a general sensor clock reading and does not need to equal any actual reading $n(i)$. Eq. 5 represents the result of time synchronization as considered in the presented work, a function that transforms readings of time relative to the sensor clock into time relative to the reference clock.

Note that $n_s$ must be an integer, since $t_{tick}$ represents the resolution of the sensor clock, and cannot be further divided. For the same reason, the sensor clock frequency defined by the clock resolution must also be an integer multiple of sampling frequency. Also note that so far all the equations apply only to the ideal case with perfect clocks.

### C. Variable clock frequency

The first challenge in time synchronization is the variable clock frequency, which affects both the sensor nodes and the master device. This variability arises from fluctuations in the respective clocks, whether due to physical changes or software influences. As a result, the sampling frequency changes over time, and these variations must be accounted for.

The clock on the sensor node is often not robust and can be influenced by factors such as temperature changes or battery level fluctuations. The same applies to the master device if it is similarly a simple device like the sensor. However, the master device is typically more sophisticated (e.g., a smartphone or PC) with a more stable clock, allowing it to serve as a reference. Although generally more accurate than sensor clocks, the smart device clock is not entirely infallible; it may experience minor variations due to fluctuations in external factors such as the ambient temperature. To maintain accuracy, it requires constant synchronization with Internet time. Therefore, while the smart device clock is reliable and usually has a lower margin of error than the sensor node clock, its frequency remains subject to slight fluctuations and periodic adjustments.

Varying sampling frequency could be expressed in Eq. (5), but we avoid such formulation, since our proposed solution handles it differently. We consider the sampling frequency to be constant for some short periods of time and we formulate the equations with this in mind - equations are only applicable for short intervals.

### D. Random delay

Random delays, as explained before, can occur during wireless communication. Actual relation between reception and transmission times is represented by the blue circles in Figure 1, and can be contrasted to the ideal relation (equality) represented by black circles. Real reception timestamps (timestamps taken immediately upon reception of a wireless data packet) are higher than transmission timestamps (timestamps taken immediately before triggering a transmission) by some positive delay. This delay is composed of several parts that are constant in duration: the time it takes the sensor to pack the timestamp for transmission, the time it takes the communication packet to be transmitted between the devices, the time it takes the receiving device to unpack packets and take its own timestamps. However some parts are of variable duration and these represent the source of variability of the delay.

Initially, there are wait times in buffers on both the transmitting and receiving software, required by the operating systems or the communication protocols. Additionally and most importantly, wireless communication protocols such as Bluetooth Low Energy (BLE) do not transmit messages immediately when they become available but only during allowed time intervals, which are not known on the application layer of software and can thus not be predicted. Messages fully prepared for transmission simply wait until the first allowed time interval is reached.

To mathematically describe the effect of random delays, we introduce a random variable $\xi$ to Eq. (5). This variable represents the random delay, and as a result, the time of arrival of the $n$-th sample $t_i^{RD}(n)$ is now expressed as follows:

$$\begin{aligned} t^{RD}(n) &= t^{RD}(n) + \xi(n) \\ &= t_{start} + \frac{n - n_{start}}{n_s f_s} + \xi(n). \end{aligned} \tag{6}$$

### E. Packet loss

The packets lost in transmission are another common issue in wireless communication and can significantly impact application performance due to the absence of expected data. In the worst case, packet loss may cause some essential synchronization data to be lost. In the context of the proposed time synchronization, packet loss is generally a severe event, it only causes less data to be available for synchronization procedure, which may negatively impact the accuracy. Single packet losses with the frequency of one lost per hundreds of received packets has a negligible impact, however, acute packet loss in some time interval can cause synchronization accuracy to decrease within that time interval.

## III. Principles of Synchronization Methodology

To achieve offline synchronization, synchronization data (metadata) has to be collected in addition to the measurement data. The presented approach assumes that synchronization data is analysed only after the measurement completes. As described in subsection II-A, the minimal required data for synchronization consists of the associated timestamps from sensor and master device, and can be collected on either of the devices. Since synchronization data is collected on one of the devices, only an unidirectional data transfer is required for the purpose of synchronization and can furthermore be mixed with the data transfer required for signal samples collection or other data transfers. While not required by the method, synchronization data is usually collected at predetermined intervals, such as every second, or with every transmitted data packet. There are no further requirements imposed on the synchronization data.

In subsection III-A we shall first focus on the case where synchronization data is included in every packet of signal samples and is collected by the master device. In subsection III-D

we shall explain the difference with the case where sensor devices collect synchronization data instead, and a dedicated wireless data channel is established for synchronization data.

Synchronization starts with the procedure in which clocks of sensor nodes are aligned with the reference time; the timestamps for samples are translated from the tick counts of the local sensor clock to reference clock, represented by the master device's clock and expressed in seconds. This translation is called time alignment and it alone addresses all the presented synchronization challenges (random delay, variable sampling frequency, and packet loss), the details of which are described in subsection III-B.

Next, it is assumed that the measurement is valid only when data is available for all the sensors. Therefore, a common time interval is selected in which data from all sensor nodes is available and data collected by some sensors either before or after this interval is discarded. The obtained time interval is then sampled with the desired final sampling frequency to get equidistant timestamps, as described in more detail in subsection III-B.2.

Finally, the signals of each node are interpolated to obtain their values at the obtained timestamps and generate resampled signals. This is the final result of our proposed method and is described in subsection III-C.

At this point, the accuracy of results cannot be verified in general case. However, experiments can be designed to produce highly correlated data on multiple sensors. Synchronization can be verified on the measurements from such an experiment using cross-correlation, with a procedure described in subsection III-E.

### A. Example applications

In this paper, we present two different example applications, one with ECG and one with IMU sensors. A photograph of sensor nodes is presented in Figure 3. Regarding the synchronization procedure, they differ in the direction in which the synchronization data travel. In the first example application, timestamps are sent from the sensor nodes (ECG sensors) to the master device (Android smart phone), while in the second, timestamps are sent from the master device (Android smart phone) to the sensor nodes (IMU sensors).

For the ECG example, we use a system described in Rashkovska et all [30], comprising a combination of the Android application MobECG and a set of Savvy ECG sensors. The properties of the latter are: the input is a single-channel voltage in the mV range, the sampling frequency is 128 Hz, a BLE connection is used for communication, and the communication packets are not acknowledged when received, resulting in possible data loss. The primary communication the devices is triggered by the sensor node and comprises communication packets containing one timestamp and 14 samples. The packets are created after a set of 14 samples is taken and forwarded to a module that handles BLE communication. The timestamps of the sensors correspond to the counters of the samples. They start with 0 at the beginning of the measurement and then increase by 14 with each packet. Missing packets are detected by observing differences in the consecutively received

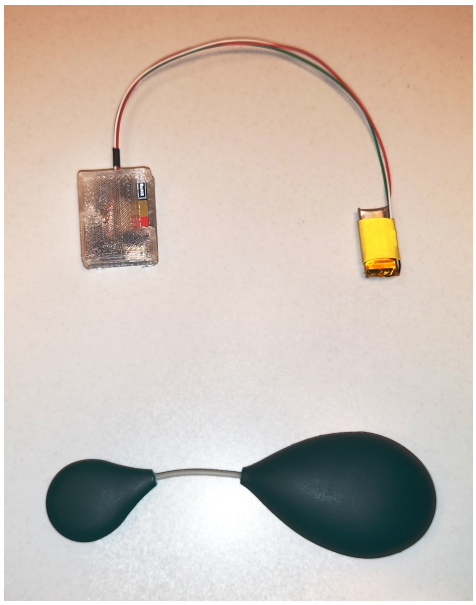


Fig. 3. Two sensor node types used in the study. On the top is the IMU sensor node in a 3D printed case with and battery separated by long wires, to be attached to the body separately to lower the sensor's mass. Below is the ECG sensor in its two-part injection molded case, where each part of the case houses one of the electrodes, while the larger part also houses the battery and electronics. The ECG sensor measures about 12 cm in length.

timestamps. The frequency of the sensor clock characterized by sample counter corresponds to the sampling frequency and is 128 Hz.

For the IMU example, we use an experimental Android application and an experimental IMU sensor based on STM SensorTile, both of which are part of a research project. This sensor samples three axes of linear acceleration, three axes of angular acceleration and three axes of magnetometer and records them locally on an SD card. The sensor's firmware is based on FreeRTOS and uses the kernel-based clock with a resolution of 1 ms. The communication between the devices is one-way, always from the smartphone to the sensor, and contains either metadata for the sensor, which is recorded with the measurement, or timestamps with nanosecond resolution for offline time synchronization. The latter is triggered regularly with a frequency of 1 Hz.

The ECG example and its direction of communication serves as the base for the presented method in the following sections.

### B. Time alignment

The fundamental step in data synchronization involves mapping the time recorded by each involved clock to a single reference time. This phase of synchronization has the largest effect on the accuracy of the final synchronization of signals within a single measurement.

The input consists of pairs of timestamps from the sensor clock and the reference clock. The sensor clock is a monotonically increasing integer counter, denoted as $n(i)$, and the reception time of the reference clock, i.e., the reference timestamp, is denoted as $t^{RD}(i)$. The sequence of tuples

$\big(n(i), t^{RD}(i)\big)$ is valid for $i \in [1, N]$, where $N$ corresponds to the total number of obtained time synchronization tuples in the measurement. Typical relation between the timestamps is plotted in Figure 4. Time alignment is applied individually for each sensor data in three steps explained below.

*1) Random delay:* Random delay is the first challenge to be addressed. To achieve time alignment, linear regression [31] is first used to fit a linear function to the input data tuples. The result is the predicted reference time $\hat{t}(n)$, which acts as an approximation to the true but unobtainable $t(n)$ from Eq. 7, and is shown as an orange linear function in Figure 4.

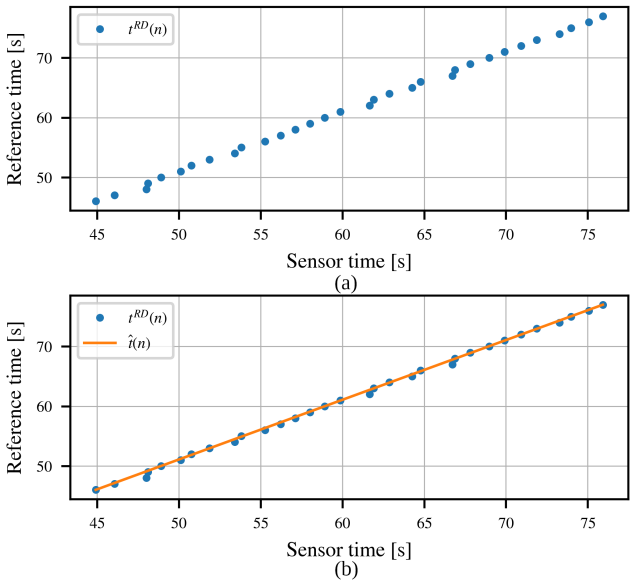



Fig. 4. A step of linear regression performed on a real measurement. Range of the displayed time was manually selected. (a) Timestamps as received by the device. (b) Reference time prediction with linear regression

Although Figure 4 seems to indicate that linear regression deals effectively with the random delay, it is actually not precise enough, and errors in the order of milliseconds remain. To see the errors in more detail, a different visualization is required. In Eqs. 5 and 7, variables $t(n)$, $\xi(n)$ and $f_s$ can be substituted by their approximations $\hat{t}(n)$, $\hat{\xi}(n)$ and $\hat{f}_s$. The resulting equations:

$$\hat{t}(n) = \hat{t}_{start} + \frac{n - n_{start}}{n_s \hat{f}_s}, \tag{7}$$

$$t^{RD}(n) = \hat{t}_{start} + \frac{n - n_{start}}{n_s \hat{f}_s} + \hat{\xi}(n), \tag{8}$$

can then be combined to express random delay as the difference between estimated and received reference timestamp:

$$\hat{\xi}(n) = \hat{t}(n) - t^{RD}(n). \tag{9}$$

The delay is visually informative, so we plot it to verify the effectiveness of the time alignment procedure. We show the delay after several steps of the proposed method in Figure 5. Top subfigure shows the delay after the first step, i.e. the linear regression. To be able to analyse the calculated delay, we list the physical properties that the random delay should exhibit:

- It cannot be negative, since the timestamp from reference clock cannot be received before it has been obtained; therefore its lower bound is 0.
- Its upper bound is unknown, but may be almost arbitrarily high since the conditions for wireless transfer may be arbitrarily bad.
- The conditions for wireless transfer can vary with time, therefore the delay is not expected to be stationary.

From the delay just after linear regression in Figure 5, we can see two issues. Time alignment by linear regression creates the impermissible negative delays, and a visible trend in delays. The latter is caused by the non stationary input data which manifests as the non stationary resulting delay (much higher variability of the delay at the start than at the end of the measurement).

To improve the results of linear regression, i.e., remove the mentioned issues, we perform a non-linear optimization as an iterative procedure described below and demonstrated in Figure 5. The subject to optimization are variables $\hat{t}_{start}$ and $\hat{f}_s$. Input to $i^{\text{th}}$ iteration of the procedure is the $i$-th linear transformation:

$$\hat{t}_i(n) = \hat{t}_{i,start} + \frac{n - n_0}{n_s \hat{f}_{i,s}}, \tag{10}$$

with the initial values (that is $\hat{t}_{0,start}$ and $\hat{f}_{0,s}$), obtained from the linear regression as described above. The iterative procedure is executed until the linear transform does not change within a single iteration:

1) Calculate delays $\xi_i(n) = \hat{t}_i(n) - t^{RD}(n)$ (Figure 5).
2) Find *pivot samples* $\xi_{1/3}$ and $\xi_{3/3}$ of the measurement, i.e. samples with lowest delay $\xi$ in the 1st and 3rd thirds of the measurement, respectively (green dots in Figure 5). Then calculate the required adjustment of current iteration represented as the linear coefficient of the line that connects the two pivot samples: $k_{i,e} = \frac{\xi_{3/3} - \xi_{1/3}}{t(\xi_{2/3}) - t(\xi_{1/3})}$ (Black line in (Figure 5)).
3) Fix the estimated sampling frequency in the equation for the linear transformation: $\hat{f}_{i+1,s} = \hat{f}_{i,s} + 1/k_{i,e}$.
4) Eliminate negative delays by offsetting the linear transformation: $\hat{t}_{i+1,start} = \hat{t}_{i,start} + \min(\hat{\xi}_i(n))$

The resulting delays after the last step are plotted in Figure 5 to demonstrate the achieved improvement of single iteration of the proposed algorithm. Note that although not visible in Figure 5, after one iteration, the delays may not be strictly positive and some trend may remain in the delays.

*2) Variable sampling frequency and packet loss:* After dealing with the random delay, the packet loss and variable sampling frequency need to be addressed. We minimize the effect of both these challenges by means of decomposing a long measurement into short blocks and applying time alignment on each block separately.

On a long enough time scale, the relation between the two clocks is not linear, since both clocks experience substantial drifts. A demonstration of the resulting long-term relationship between clocks expressed in form of the delay after applying linear transformation is shown in Figure 6. Although a simple linear relation is insufficient to fully explain it, the observed

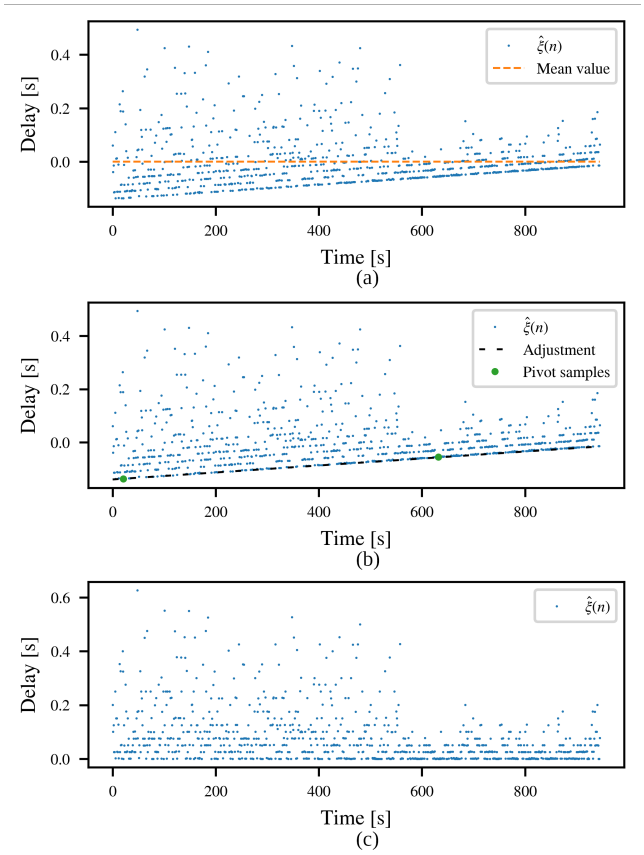



Fig. 5. Estimated reference clock delay as it changes throughout the one iteration of the non-linear optimization. (a) Delays calculated in step 1 (as in Figure 4). (b) Pivots and adjustment calculated in step 2. (c) The results of steps 3 and 4 of iteration visualized as delays calculated in the next iteration.

relation is not overly complex and a piece-wise linear function can express it with sufficient accuracy. To prepare for a piece-wise linear approximation, the synchronization tuples are partitioned into blocks with time alignment performed on individual blocks instead on the whole measurement.

The process involves dividing the measurement into $M$ blocks, denoted as $B_m$, where $m = [1, M]$. Each block covers a limited range of input synchronization tuples: $B_1 = [b_0, b_1]$, $B_2 = [b_1+1, b_2]$, ..., $B_M = [b_{M-1}+1, b_M]$, where $b_0 = 1$ and $b_M = N$. In addition to serving as a base for the piece-wise linear function construction, blocks also serve in eliminating some of the packet loss. Significant number of lost packets from a single sensor within a short time frame could introduce errors if used in the time alignment procedure. Therefore parts of measurement with significant number of missing packets are marked as bad blocks and later skipped in time alignment.

In the ECG example application presented here, packets of wireless transmission comprise timestamp and 14 ECG samples, loss of a packet incurs loss of both, a timestamp and ECG data. Since in this example, data with frequent intervals of missing samples are not fit for analysis, they are discarded. Skipping bad blocks is thus not problematic, time alignment is skipped only on intervals where data is also discarded. A different approach might be needed in cases where all the recorded data should be retained. An option would be extrapolating data from neighbouring blocks or allowing bad blocks to cover longer temporal interval, but regardless of the approach, the accuracy of the achieved synchronization would necessarily be lowered for bad blocks.

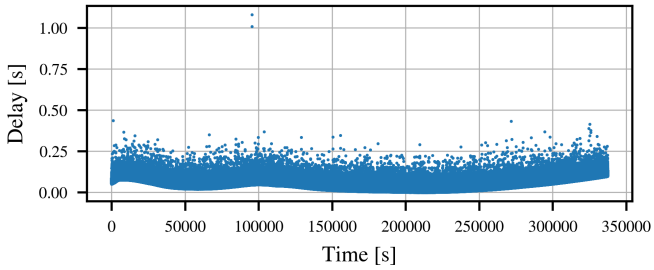



Fig. 6. Delay calculated from the reception and transmission timestamps with the assumption of linear relation between clocks. Random delay in packet reception is visible as well as clock long-term drift (local minima in delays vary with time).

Two processes are used for decomposition into continuous blocks: quality decomposition and frequency decomposition. These methods ensure that the blocks consist of contiguous and uninterrupted data that is free of packet loss.

**The quality decomposition** process focuses on partitioning the synchronization tuples into good and bad blocks. To cluster synchronization tuples into blocks, the procedure assigns packet 1 to the first block. This block is now considered open, since its lower bound is set (to 1), but upper bound is yet unknown. Then, iterating through synchronization tuples in order, the procedure examines each pair of consecutive tuples $i$ and $i+1$, and calculates the time interval between them (we shall name it $\Delta t(i, i+1)$). If $\Delta t(i, i+1)$ is greater than a pre-defined parameter $t_{pause\_max}$, then packet $i$ is set as the upper bound of the open block, and the block is thus closed. New block is opened with lower bound set to $i+1$. If $\Delta t(i, i+1) < t_{pause\_max}$ nothing happens and the procedure continues. After the last packet has been processed, the last open block is closed with it upper bound set to packet $N$.

At this stage, blocks are identified but unmarked. To separate bad blocks from good, we introduce a new threshold value, $t_{good\_min}$ as the minimum length of a good block. Block $B_m$ with bounds $[b_m - 1, b_m]$ is therefore considered good if $\Delta t_{(b_{m-1}, b_m)} \geq t_{good\_min}$; otherwise it is marked as bad.

The second process is **frequency decomposition**, and its task is to subdivide the previously obtained good blocks into shorter blocks of a specified length, denoted as $t_{target}$, which should be approach the maximal time in which the estimated clock drift can be considered linear for the selected hardware. In this process, each good block longer than $t_{target}$ is subdivided into $\lceil t_{block}/t_{target} \rceil$ blocks of approximate length $\frac{t_{block}}{\lceil t_{block}/t_{target} \rceil}$ (tuples with nearest receipt time are used as boundary points). The resulting good blocks are appropriate for piece-wise time alignment, which is able to follow the clock drifts.

Selection of values $t_{pause\_max}$, $t_{good\_min}$, and $t_{target}$ should be done empirically, relying on trials conducted on measurements with noticeable irregularities. Generally, they should be setup to balance maximizing block length for optimal time

alignment with limiting block length to capture changes in clock frequencies. In our ECG example, we used the following values: $t_{pause\_max} = 1\ s$, $t_{good\_min} = 10\ s$, and $t_{target} = 1800\ s$.

After partitioning completes, the time alignment is performed for good blocks and is skipped for bad blocks to obtain a piece-wise linear clock transformation function.

*3) Sampling time estimation:* So far, the proposed method worked with synchronization tuples only – which are fewer in number than the sensor sample packets. In this step, the obtained piece-wise linear transformation is applied on all the data samples that are covered by good blocks. When this step is complete for all sensor nodes comprising the measurement, the obtained aligned timestamps are all relative to a common clock, and can be considered synchronized.

### C. Resampling

An additional, optional step may be applied depending on the application requirements. In some cases, it is necessary for the signals originating from various sensors to be sampled at same times, which is not inherently achieved after time alignment. The only way to create a common time axis for all signals in a measurement is through signal resampling. With resampling, first the values for sampling times are calculated using a common time interval for all sensors and for a desired resampling frequency. Various resampling methods exist [32], differing primarily in the interpolation techniques used to estimate the signal value between existing samples. In the presented study, we use a spline interpolation methods with cubic polynomials.

### D. Alternative example application

The procedure described above refers to the ECG example application, but there are several important differences for the IMU example application. The first is that from the set of described time anomalies, only random delays are observed in the IMU case. The reason is that the measurements in this example application are short, lasting from several minutes to half an hour. As such they do not need to be divided into blocks and all the available synchronization data can be used for the time alignment. Although we detected no cases of packet loss in IMU example, which might be because of the laboratory setting, some packet loss would be tolerable just as it was in the ECG example.

The second difference is that the direction of wireless data transmission; it is reversed, compared to the ECG example. In the IMU example, the sensor nodes receive reference timestamps from master device, and in such a case the reference timestamps $t(i)$ are taken before sensor timestamps $n(i)$. Because the timestamp on sensor node is recorded after the reference timestamp, the order of terms in the delay definition needs to be reversed:

$$\hat{\xi}(n) = t^{RD}(n) - \hat{t}(n). \tag{11}$$

This is the only change required in the presented method.

Note that in our IMU example, aside the synchronization metadata, also the signal samples are stored on the sensor nodes instead of the master device. This does not play a role in the synchronization, however, it does require one to upload the signals from all the sensor nodes to a common device before attempting synchronization.

To summarize, the only requirement is that the definition of delay is positive, regardless of the direction in which the synchronization tuples travel. Therefore, the time alignment and with it the whole synchronization procedure are applicable either if the timestamps that require alignment are taken before or after the reference timestamps, which covers all possible cases of time synchronization with simple synchronization primitives consisting of only two timestamps.

### E. Verification of time alignment

To evaluate the accuracy of the time alignment, it is usually necessary to compare the aligned signal with a known signal value. The cross-correlation between signals can be used as a tool for comparison [33]. Suppose that all the data has been resampled at time points $t_1, t_2, ..., t_m$, the first sensor records values $p_1, p_2, ..., p_m$, and the second sensor records values $q_1, q_2, ..., q_m$. To compare signals, the task is to determine the displacement $k \in \mathbb{Z}$ that maximizes the cross-correlation, calculated as $p_1 q_{1+k} + p_2 q_{2+k} + ...$ Since the sampling time $t_{sampling}$ – the difference between successive time points, is defined by the resampling frequency, we can calculate the error of synchronization as the apparent delay between the two signals: $k\ t_{sampling}$. Accuracy can then be considered the absence of error. To track the accuracy as a function of time, one can perform cross-correlation not on the whole measurement but rather on a rolling window to get estimates of local errors.

Since we do not have a possibility of generating a signal synchronized with the reference (smartphone) clock in our example applications, we do not posses the means to determine errors in the absolute sense, as defined above. We use relative errors instead, which means that we cross-correlate signals from two separate sensor nodes. Mathematically, relative errors between synchronization results $a$ and $b$ are the difference between absolute errors for $a$ and $b$. As a combination of two components, relative errors may either mask or amplify the components making them inferior to absolute errors in this respect. However, relative errors are actually a better measure of the synchronization accuracy for the task that the proposed method was designed for. That task is preparing signals from different sensor nodes for further analysis that requires them to be synchronized between each other, and not to an absolute time.

Both of the example applications use three sensor nodes, which creates three possible pairs of synchronization results for comparison. Although the third error can be calculated by subtracting the other two errors, we analyse all three errors to make their spread clearer in visualizations.

## IV. Results

In the Results section, we present the experimental setup, experiments, and results of the proposed synchronization

method for the two previously defined examples. The approaches to synchronization differ between the examples, since the examples themselves are quite different from each other. In the first example, timestamps are sent from the ECG sensor to the phone. Measurements are long and synchronization faces all three challenges mentioned earlier, i.e., random delay, packet loss, and variable sampling frequency. In the second example, on the other hand, timestamps are sent from the phone to the IMU sensors, measurements are short, and synchronization faces only random delay in transmission.

### A. Experimental setup

To generate ECG and acceleration signals, we use simple generator setups with readily available components based on Arduino (a development board that houses 8-bit microcontroller intended for rapid prototyping). More specifically, the ECG signal is replicated by an Arduino-based signal generator that approximates the ECG by transposing multiple short sine waves on a DC carrier signal, with the patter repeating periodically once per second. The generator uses only resistors and capacitors in addition to the Arduino board. This generator is subject to some clock drift and its clock is not synchronized with that of the sensors or the smartphone that records measurements, so the true timing of the ECG signal is unknown but is also not required. All three sensors are connected to the same signal source and the same Android device via a Bluetooth LE wireless connection (Figure 7).

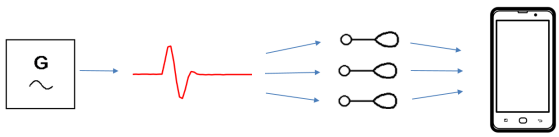



Fig. 7. A scheme for simultaneous artificial ECG signal measurement with 3 sensors.

The IMU measurement was performed with three IMU sensors that were physically connected to each other and formed a rigid package. The package was placed on a device that applied short acceleration pulses controlled by the Arduino. The pulses were generated once every 10 seconds by rotating a servo motor arm in one direction for 50 milliseconds and then back to its starting position. The servo motor that was used did not reach its designated rotation in such short time interval but it did produce maximum accelerations that it was capable of. Similarly to the ECG example, the IMU pulse generator is subject to some clock drift, its clock is not synchronized with that master device, and the true timing of the IMU signal is unknown. The linear acceleration in the axis of the pulses was extracted from the measurement to test the synchronisation of the sensors.

Since both of our example applications (ECG measurement and gait measurement with IMU sensors) require the use of three concurrent measuring sensors, we replicated the use of three sensors in our laboratory configuration. Thus we have three concurrent samples available in both cases, which we can use to estimate the accuracy of the time alignment method.

### B. Synchronization of time in wireless ECG measurement

The following results show the evaluation of the proposed methodology for the experiment with the ECG example.

A 93-hour concurrent measurement was performed with three sensors. 93 hours was actually the time in which the first sensor depleted its battery while the remaining two sensors would be able to continue measurement for several more hours. All signals were received by a single Android smartphone where they were stored separately and later transferred to a personal computer for processing. To determine synchronization accuracy as a function of time, the synchronization error was calculated once per second of measurement (i.e. for each interval of the artificially generated ECG signal) for all three pairs of synchronization results.

*1) Before synchronization:* In Figure 8, the raw signals of the measurement are shown. About 250 ms of the signal from all three ECG sensors is shown after more than 77 hours of signal measurement. This particular interval was selected for visualization, since the largest synchronization errors were measured on it. The time offset between the signals is obvious from the figure and illustrates the need for synchronization in multi-sensor ECG systems to ensure accurate analysis.

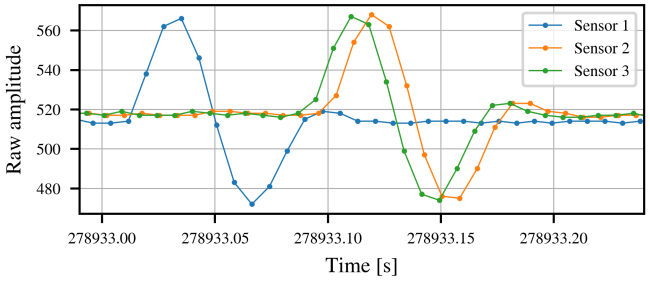



Fig. 8. Raw unmodified signals from the ECG sensors. The signal of the first sensor is shown in blue, that of the second in orange and that of the third in green.

The graphical representation of the signals takes into account the last received timestamp before the start of the interval in focus. It is assumed that the sampling is uniform, based on a predetermined default frequency. We observed this approach to typically show a delay of 50 to 100 milliseconds between the signals. This latency is characteristic of BLE technology, which is largely responsible for the time discrepancies observed in the described plotting technique. If we observed only the received signal without the reference timestamps, we would find that after 77 hours of measurement, sensor clocks have differed by about 1.3 s.

*2) Time alignment:* After data acquisition, synchronization was performed as described in Section III. In the first phase, the time alignment was performed. In Fig. 9 the ECG signals after individual time alignment are presented. It can be seen that the synchronization between the various sensors has been improved and is now $\approx$ 2 ms between Sensors 1 and 3 for the interval shown, while Sensors 2 and 3 are almost perfectly synchronized.

*3) Resampling:* In Fig. 10 the signals from all three sensors are shown after time alignment and resampling with cubic

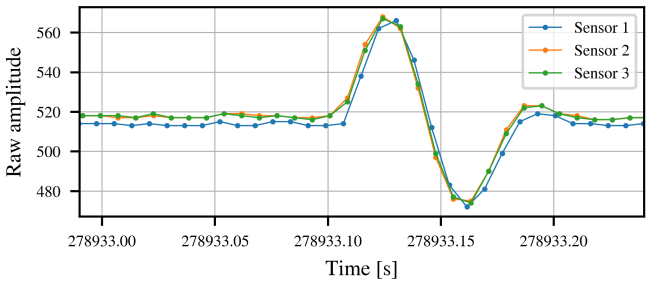



Fig. 9. Signals from ECG sensor after individual time alignment.

spline interpolation. There is no further improvement in synchronization at this point of the procedure, signals have only been resampled with frequency of 2000 Hz to aid verification process. In a real use case, eg. as described in [9], [34], the signals would potentially be resampled with a frequency similar to the original sampling frequency and then linearly combined.

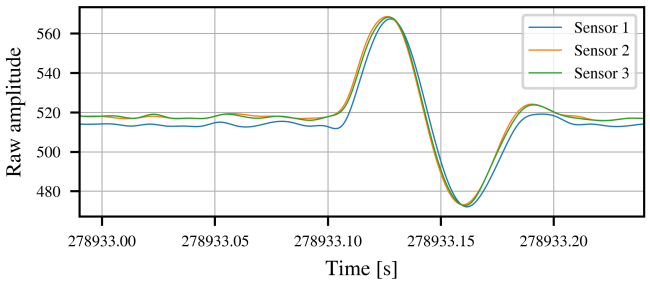



Fig. 10. Aligned ECG sensor signals after cubic spline interpolation and resampling.

*4) Verification:* Figure 11 (a) shows a histogram representing the distribution of synchronization errors shown in milliseconds, for the three sensor pairs. The horizontal axis indicates the magnitude of the synchronization error, while the vertical axis indicates the absolute frequency of these errors within the data set. The three synchronization errors are color-coded. The original sampling time is represented by a dashed line and serves as a reference against which the synchronization of the sensors is evaluated. The frequency distribution is strongly skewed towards the lower end of the synchronization error spectrum, with a pronounced peak at the beginning of the spectrum, indicating that minimal synchronization deviations close to zero milliseconds prevail for all sensor pairs. It can be observed that the absolute frequency of synchronization errors is mostly within a 1.5 millisecond. The spread of errors indicates where the limitations of the presented time alignment method lie. Errors so much lower than the sampling time are acceptable for the ECG example application. The offset of errors from zero, however, can be explained only by an external limitation of the method. We traced this offset and found it originates in the files produced by the recording software on Android. This software records signals from the connected sensor nodes individually and stores the initial timestamp, i.e. the sensor start time, which is different for each sensor node, in millisecond resolution. Then it stores all further timestamps relative to the measurement start time in microsecond resolution. Thus, the remaining offset equals is in the order of magnitude defined by the sensor start time, that i 1 ms. We speculate that the accuracy of synchronization would be further improved if the stored resolution of start time was increased.

Figure 11 (b) shows the errors of time synchronization as a function of time. The horizontal axis represents the time over the entire measurements of 93 hours. While the graphs show similar information as Figure 11 (a), they should also demonstrate any temporal trend or pattern. For this example, the graphs show no visible trend or pattern over time, the distribution of errors seems stable over time. The limit of error can be seen from this plot more clearly and is at the 2.2 millisecond mark.

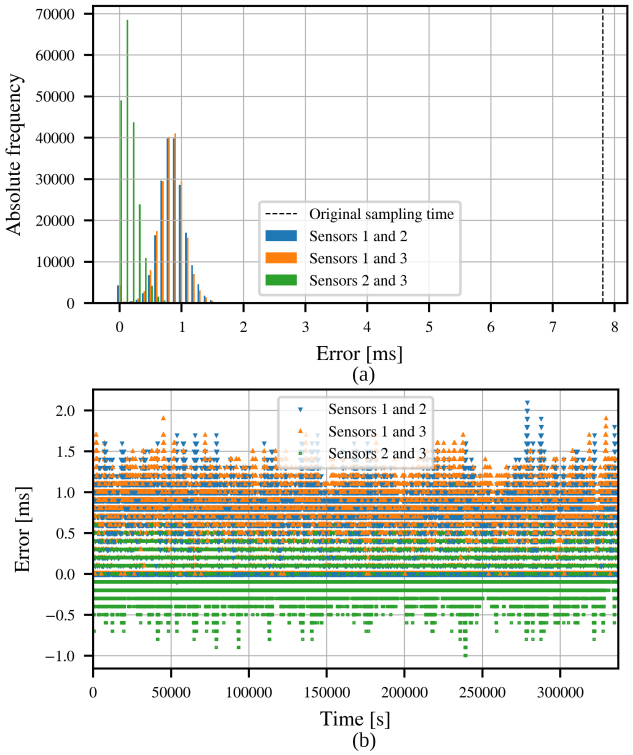



Fig. 11. Accuracy of time alignment on the ECG example. Verification is performed by cross-correlation between signals around individual peaks in the data. (a) Distribution of errors. (b) Absolute error as a function of time.

To illustrate the challenging nature of the input data for ECG example we also analyse missing packet statistics. Wireless transmission packets cannot be received out of order in this example, causing the timestamps of received packets to be monotonically increasing. Missing packets can thus be identified by detecting the anomalies in the difference of sequentially received packets. The gathered statistics of missing packets is shown in Figure 12 The amount of missing data is about 0.02% for sensor 1, 10% for sensor 2, and 3% for sensor 3. On one hand, up to 10% of data loss seems quite high for a laboratory setting, but on the other hand, the experienced data loss was not concentrated enough for any part of any measurement to be marked as a bad block. Therefore in the presented case, the algorithm managed to classify the

whole measurement as good and then perform time alignment on all of its parts.

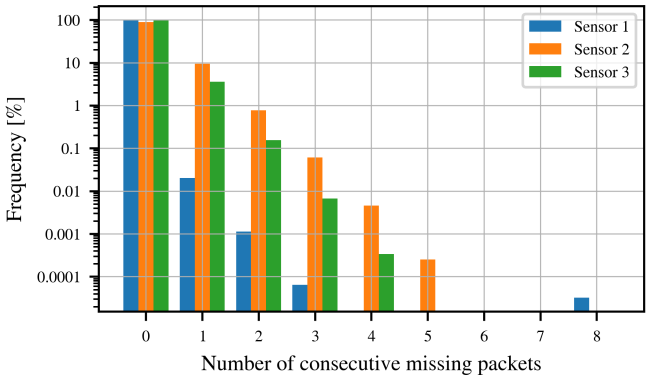



Fig. 12. Statistics for the missing packets in ECG use-case. Longer sequences of missing packets present can lead to loss of accuracy in time alignment.

Data verification was performed only on the parts where measurement was valid on the compared 100 ms interval by all three sensors. Although individual sensor measurements contain 90% of valid samples or more, the compound criterion for data verification meant that comparison could be made only on 40% of the 337471 recorded peaks. Despite expecting the laboratory setting would provide for a larger portion of data to be available for analysis, we find this value large enough to be confident about the results.

### C. Synchronization of time in wireless IMU measurement

The following results show the evaluation of the methodology described above for the IMU sensors. An 1 hour concurrent measurement was performed with three sensors, which is in line with capabilities of the sensor nodes but is above the aimed 20 minutes long measurements for the real-life sensor use. Each signal is recorded at the sensor node, stored on the SD card, and later transferred to a personal computer, where it undergoes synchronization processing.

Synchronization tuples are generated differently than in the ECG example. Android smartphone app connects to the sensor nodes, orders them to start sampling and then periodically, every second transmits local time to the sensor nodes. All communication is point-to-point and synchronization timestamps are taken separately for sensor node to pe as precise as possible. As the sensor node receives synchronization packet, it takes local timestamp, creates synchronization tuple and stores it along the sensor data on the SD card.

In this example application, the typical measurement time is too short for clock drift to be significant and the primary timing challenge is the random delay. Although not as complex as the first application, addressing random delays is still important to ensure accurately synchronized data for effective gait analysis.

The measurements were performed with a sampling frequency of 500 Hz. All three axes of acceleration from one sensor are shown in Figure 13. As the figure shows, the acceleration values vary considerably along the different axes. Z-axis was the one that was acted on by the generator and is therefore the only one further analysed, shown in all further figures, used for peak identification and synchronization verification. The synchronization error was calculated once every ten seconds around each detected peak.

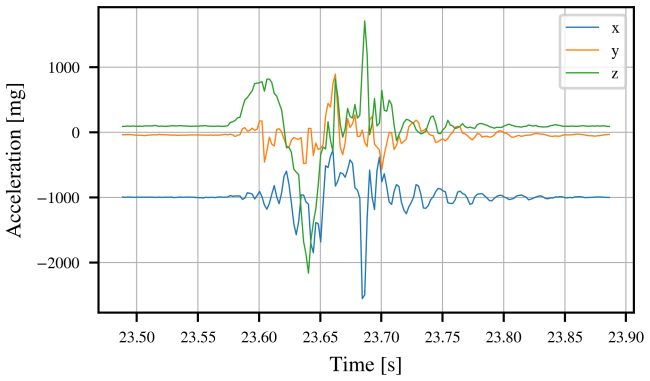



Fig. 13. Three axes of the acceleration readings around one of the identified peaks in data from the IMU use-case.

*1) Before synchronization:* Fig. 14 shows the original acceleration signals of the z-axis before synchronization. After 23 seconds of signal measurement, about 40 ms long portion of signals from all three sensor nodes are visible. Date was plotted in the same way as ECG example above. Similar to the ECG signals, a clear time offset can be seen between the signals from all three sensors, indicating the need for synchronization.

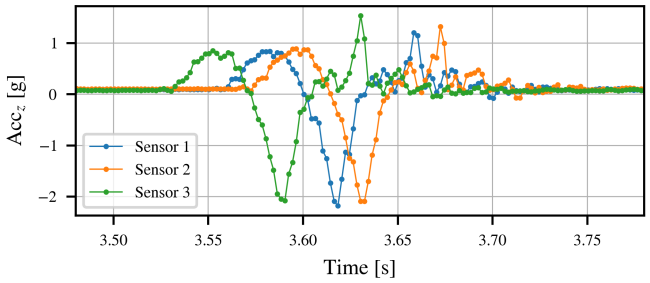



Fig. 14. Raw unmodified signals from the IMU sensors. The signal from the first sensor node is shown in blue, from the second in orange and from the third in green.

*2) Time alignment:* After the data acquisition was completed, the synchronization of the recorded signals was performed as explained in Section III. In the first phase, the time alignment was performed without partitioning into blocks. Fig. 15 shows the IMU signals after the individual time alignment. Sensor 1 and Sensor 3 are in almost perfect alignment at the shown moment in time, while signal from Sensor 2 deviates slightly.

*3) Resampling:* Figure 16 shows the signals of all three sensors after resampling with cubic spline interpolation at 5000 Hz. No additional improvements in synchronization are expected from this procedure, but the synchronization errors are visible better and the resulting signals can be used as input to verification procedure.

*4) Verification:* Figure 17 is plotted in the same way as Figure 11. Figure 17 (a) shows a histogram representing the

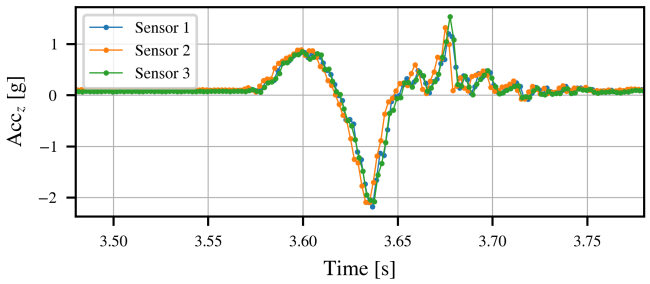



Fig. 15. Signals from IMU sensors after individual time alignment.

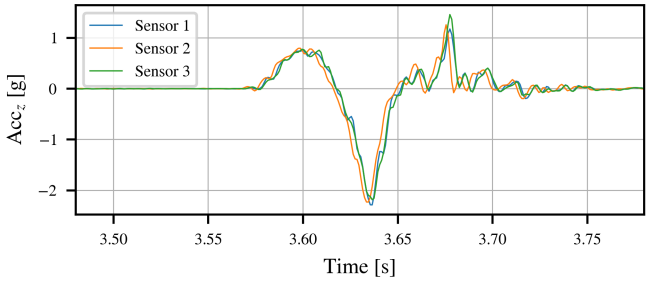



Fig. 16. Aligned IMU sensor signals after cubic spline interpolation and resampling.

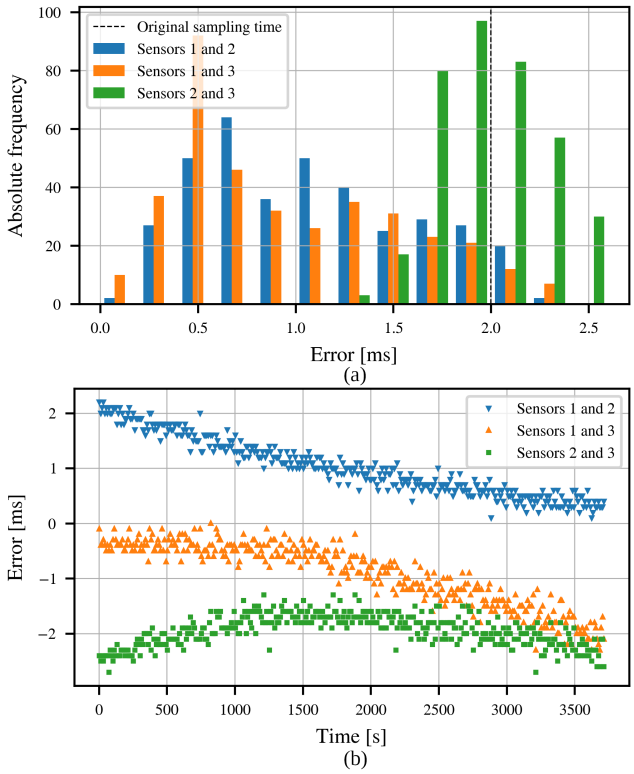



Fig. 17. Accuracy of time alignment on the IMU example. Verification is performed by cross-correlation between signals around individual peaks in the data. (a) Distribution of errors. (b) Absolute error as a function of time.

distribution of synchronization errors in milliseconds, for the three pairs of sensors. Again, the original sampling time is represented by a dashed line. The magnitude of error is about the same as in ECG example, however, since the sampling time of the IMU example is much lower, synchronization error is sometimes higher than sampling time.

Figure 17 (b) shows errors of time alignment as a function of time for the three pairs of sensors. Unlike in the ECG example, patterns are visible in the synchronization errors and the spread of data points indicates the method is quite precise but not accurate. I.e., the errors obtained on a single pair of sensors are grouped closely together but are not always close to 0. The offset from zero is again – similarly to the ECG example – partly caused by using timestamps with 1 ms resolution on synchronization packets. The reason for the 1 ms resolution is different however, this time it is due to the clock on the sensor only using 1 ms resolution. Error is aggravated by the lack of synchronization data – only one synchronization point per second is available while in ECG example there is a synchronization point for every 14 samples, which sums to more than 9 per second at sampling rate of 128 Hz. The resulting errors are also not linear with time, indicating that sensor clock frequencies changed unevenly during the measurement and piecewise-linear time alignment would be required to eliminate this effect. Since the number of synchronization points is so low, however, such an approach would make even less points available for each linear section which would likely result in higher errors. Avoiding the partitioning into blocks is also in line with the intended use of the IMU sensors however, which defines measurement times to be only 20 minutes or less, which is 3 or more times less than in our experiment.

## V. Discussion

The main focus of our study was on the inherent challenges of time synchronization in the context of wearable sensor applications. The decision to employ offline synchronization was driven by its suitability for the specific challenges and conditions of our use cases.

The proposed method provided us with the flexibility to meticulously analyse and correct time-related anomalies post data collection, ensuring the highest degree of precision and reliability in our findings. Through a careful investigation of two different use cases – electrocardiogram (ECG) and inertial measurement unit (IMU) monitoring — we aimed to explore the effectiveness of our proposed solutions and understand the limitations of existing technologies and methods. Our research focuses on not only measuring the precision of time synchronization, but also understanding the various factors that contribute to this precision.

During our research, we have come across three distinct time anomalies that greatly impact the synchronization of time in wearable sensor networks: random delays, varying sampling rates, and packet loss.

*Random delay.* The use of linear regression to apply a linear transformation to the input data to correct for random delay represents a significant step in our synchronization process. However, its inability to handle non-stationary input data required further refinements. By implementing a non-linear optimization for the linear coefficient and offset correction, we achieved satisfactory accurate time alignment.

*Variable sampling frequency and packet loss.* The challenges posed by both packet loss and a changing sampling frequency were addressed through the refinement of linear relation between clocks into piece-wise linear relation. By partitioning a long measurement into smaller, more manageable blocks and performing time alignment for each block, we were able to effectively combat the effects of both challenges. Our approach, which includes quality and frequency decomposition, ensures that the resulting synchronization remains accurate in long measurements where clock frequencies drift considerably.

After conducting a review of both example applications, we discovered that the presented synchronization method is likely more accurate than the results demonstrate. The limits of synchronization accuracy are most likely imposed by the limited accuracy of the synchronization timestamps. Both example systems could be redesigned in software and firmware to improve synchronization accuracy.

Starting with the more accurate of the two, the ECG example application, we examine the properties of timestamps taken by the master device and the ECG sensor. Sensors work with clock resolution equal to the sampling frequency, which is 128 Hz in the presented case. Yet this resolution is not the limiting one, since timestamps are generated with deterministic procedure after a clock tick. While the length of this delay is unknown, its deterministic nature makes it uniform across the sensor nodes and across all taken timestamps, with precision nearly equaling the precision of sensor hardware clock. The limiting resolution is rather the one of the reference clock taken by the master device. Measurements from different ECG sensors start at slightly different times, due to measurement start being communicated to them in series, thus they each have different start time assigned. Master device software then limits the synchronization accuracy to milliseconds by storing only millisecond resolution for start times.

The IMU example application demonstrated lower accuracy compared to the ECG example. Timestamps in this example are handled differently, with the reference timestamps from master device received on the sensor and assigned the sensor clock timestamp. Opposed to ECG example application where taking timestamps is deterministic, the process of receiving communication packets is not deterministic relative to the sensor clock, therefore the sensor clock resolution limits the precision. The IMU sensors' clocks operate and are recorded with 1 ms resolution, while the reference clock is recorded with 1 ns resolution. The lower resolution represents the more restrictive limit, thus synchronization is limited again to 1 ms accuracy. However, Figure 17 suggests that the accuracy of IMU sensor synchronization could be better than 1 ms imposed by the sensor clock resolution. There is an additional difference relative to ECG example application, which could explain the accuracy difference. Time alignment requires a large set of timestamps to operate on, and its accuracy theoretically improves as the number of available timestamps increases. The presented IMU example applications produced exactly 1, while ECG example application produces 9.143 timestamps per second (128 samples per second ÷ 14 samples per timestamp). Therefore, time alignment had 3600 timestamps available for the one hour of IMU measurement, opposed to 10971 timestamps for the 20 minutes of ECG measurement (the multi-day measurement is split into 20 minute blocks before alignment). In IMU example application, there is about 3 times less timestamps available, which should have a noticeable effect on the time alignment accuracy.

### A. Comparison to other methods

Regarding the requirements and obtained results, we can be compare our proposed method to the method by Koo et al. [19], who presented one of the most accurate approaches that can be used on wearable modules. Their GPS-based approach uses a standalone GPS module on each sensor node, relying on the modules' provided pulse-per-second signals for precise time-stamping. Synchronization of signals is performed offline from the obtained GPS-based timestamps. This method achieves a standard deviation error of approximately 40.8 nanoseconds, offering superior accuracy compared to our proposed method.

Comparing the two methods we see the following similarities and differences. The methods are similar in that they both store synchronization tuples along with the measurement and use them for offline synchronization. Both methods require processing of the timestamp tuples to obtain a piece-wise linear function that describes the relation between reference and sensor local clock.

While our proposed method requires wireless communication of timestamps, either to or from sensor nodes, method by Koo works completely locally, on the sensors. While our proposed method requires reference timestamps to be taken on master device, the method by Koo rather requires additional hardware to provide accurate reference clock on the sensor nodes. With additional hardware, additional constraints appear. The method is inherently limited by requirements such as warm-up time, a clear view of the sky for reliable GPS signals, a larger physical size of sensor nodes, and a higher power consumption. This dependency can hinder effectiveness in obstructed or indoor environments where GPS signal reception is compromised. In contrast, our method provides a more flexible solution that does not rely on any external hardware and is effective in both indoor and outdoor environments. The methods are clearly different enough to be applicable for different types of sensor nodes.

Several other approaches for synchronization of multiple sensor nodes exist, however, all depend on additional hardware or additional communication between the nodes. Approaches also differ according to the wireless technology that they are applicable on. Thus, rarely a direct comparison of different approaches can be made. The ECG example application that we worked with, however, already uses a different but compatible synchronization approach. It is implemented in software for real-time visualization of received data in Android application [30].

The comparable approach is an offline method, relies on the same synchronization data as our proposed method, however, it works in real time. Even though it works in real time, we shall not call this method online, since it does not synchronize

clocks of the sensors but rather works with their frequency drifts. The method adjusts each received synchronization timestamp immediately upon receipt through Brown's double exponential smoothing.

Brown's double exponential smoothing [35] is a time series forecasting method that is particularly useful for dealing with data that exhibit both trend and noise. The method achieves this by applying two layers of smoothing: one for the level and another for the trend. In the context of synchronizing a series of timestamps, smoothing is used to predict package submission time and transmission delay from package receive time history. By iteratively updating the smoothed values and trends, Brown's double exponential smoothing efficiently handles both short-term noise in communication delays and long-term shifts in the sampling frequency, providing a robust method for time series alignment.

Figure 18 shows the error of online (real-time) synchronization for IMU data, which can be compared to Figure 17. The results were obtained with the smoothing parameters set to 0.01 for level smoothing and 0.0001 for trend smoothing. These parameter values were obtained by a local optimization procedure with a limited number of steps applied on the shown data to avoid overfitting the values to particular data set. The results suggest that the real-time method is effective, with its errors nearing those of the proposed offline method. However, the apparent drawbacks include a long warm-up period, sensitivity to disturbances in input data (such as very high delays in received timestamps), and the inability to restrict the delay to positive values. Since the method performs time alignment by estimating the mean communication delay, it should also be susceptible to perturbances in the statistics of the delay. These do not seem to be present in the IMU example but were observed previously in the ECG example.

Figure 19 shows the results of real-time synchronization on the ECG example application, where a larger and more erratic error can be observed, compared to one obtained with the proposed method. First, a pattern with a period of 6.17 hours is visible, which can be traced back to a pattern in the statistical properties of the communication delay between the smartphone and sensors, most notably sensor number 3. We have not noticed nor analysed this pattern before, since our proposed method filtered it out very successfully. Second, the errors are not centered around 0 but at a rather large offset (relative to the sampling time). The offsets can be attributed to the mean values of communication differing among the sensor nodes.

Most of the demonstrated drawbacks of the real-time method are expected, since real-time methods can only use history to estimate communication delays, while offline methods have also future synchronization data available at each step. The emergence of pattern in error was not expected and counter-measures could possibly be taken to reduce it. Overall, the proposed method offers only minor benefits over the real-time method in our IMU example in conditions favourable to the real-time method, i.e. in complete absence of wireless connection problems. In the ECG example, however, the proposed method outperforms the real-time method, and obtains more than an order of magnitude smaller error making it more suitable for precise analysis of the collected data.

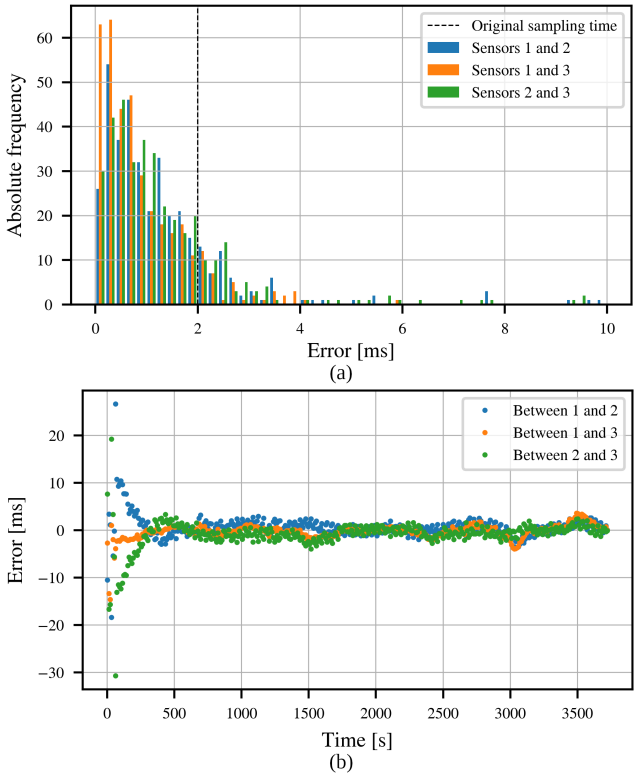



Fig. 18. Accuracy of the reference real-time sensor synchronization method on the IMU example, verified by identifying peaks on measurements and then comparing time alignment of individual peaks with correlation. (a) Distribution of accuracy. (b) Accuracy as a function of time.

## VI. Conclusion

In this study, we presented and evaluated a new approach for offline synchronization between multiple sensor nodes, focusing on ECG and IMU sensors. There are two main example applications: in the ECG application, timestamps are sent from the sensor nodes to the phone, while in the IMU application, timestamps are sent from the phone to the sensor nodes. The proposed approach can handle both cases. Within the synchronization process, challenges such as variable sampling frequencies, random delays, and packet loss must be considered. In this paper, we have detailed the methodology to address the issues related to time synchronization and demonstrated the effectiveness of our approach using different phases of the synchronization method.

## Acknowledgments

The authors would like to acknowledge the financial support of the Slovenian Research And Innovation Agency (ARIS) research core funding No. P2-0095, and project funding No. J3-3115.

## References

[1] C Wall, V Hetherington, and A Godfrey. Beyond the clinic: the rise of wearables and smartphones in decentralising healthcare. *NPJ Digit. Med.*, 6, nov 2023.

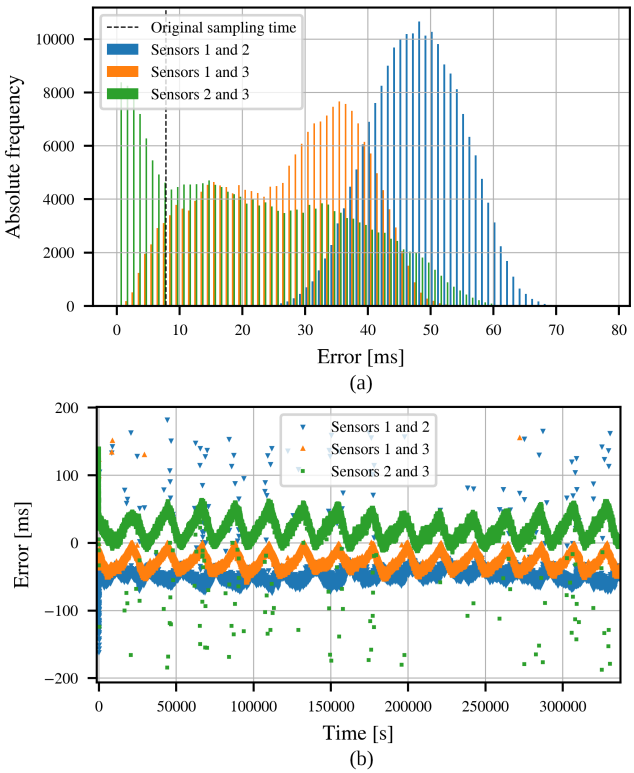



Fig. 19. Accuracy of the reference real-time sensor synchronization method on the ECG example, verified by identifying peaks on measurements and then comparing time alignment of individual peaks with correlation. (a) Distribution of accuracy. (b) Accuracy as a function of time.


[2] Huhn S, Axt M, Gunga HC, Maggioni MA, Munga S, Obor D, Sié A, Boudo V, Bunker A, Sauerborn R, Bärnighausen T, and Barteit S. The impact of wearable technologies in health research: Scoping review. *JMIR Mhealth Uhealth.*, 10, jan 2022.

[3] Jiahui Hu, Qing Qian, An Fang, Sizhu Wu, and Yi Xie. Optimal data transmission strategy for healthcare-based wireless sensor networks: A stochastic differential game approach. *Wireless Personal Communications*, 89(4):1295–1313, May 2016.

[4] WHO. Noncommunicable diseases. `https://www.who.int/news-room/fact-sheets/detail/noncommunicable-diseases`, 2023.

[5] Ramesh S and Kosalram K. The burden of non-communicable diseases: A scoping review focus on the context of india. *J Educ Health Promot*, 12:14, feb 2023.

[6] B. S. N. Alzand and H. J. G. M. Crijns. Diagnostic criteria of broad qrs complex tachycardia: decades of evolution. *Europace*, 2010.

[7] David J. Gladstone, Rolf Wachter, Katharina Schmalstieg-Bahr, F. Russell Quinn, Eva Hummers, Noah Ivers, Tamara Marsden, Andrea Thornton, Angie Djuric, Johanna Suerbaum, Doris von Grünhagen, William F. McIntyre, Alexander P. Benz, Jorge A. Wong, Fatima Merali, Sam Henein, Chris Nichol, Stuart J. Connolly, and Jeff S. Healey and. Screening for atrial fibrillation in the older population. *JAMA Cardiology*, 2021.

[8] Gloria Cosoli, Susanna Spinsante, Francesco Scardulla, Leonardo D'Acquisto, and Lorenzo Scalise. Wireless ecg and cardiac monitoring systems: State of the art, available commercial devices and useful electronic components. *Measurement*, 177:109243, June 2021.

[9] Ivan Tomašić and Roman Trobec. Electrocardiographic systems with reduced numbers of leads-synthesis of the 12-lead ECG. *IEEE Rev Biomed Eng*, 7:126–142, may 2013.

[10] Linde Ceyssens, Romy Vanelderen, Christian Barton, Peter Malliaras, and Bart Dingenen. Biomechanical risk factors associated with running-related injuries: A systematic review. *Sports Medicine*, 2019.

[11] Solvej Videbæk, Andreas Moeballe Bueno, Rasmus Oestergaard Nielsen, and Sten Rasmussen. Incidence of running-related injuries per 1000 h of running in different types of runners: A systematic review and meta-analysis. *Sports Medicine*, 2015.

[12] Yonatan Hutabarat, Dai Owaki, and Mitsuhiro Hayashibe. Seamless temporal gait evaluation during walking and running using two imu sensors. *2021 43rd Annual International Conference of the IEEE Engineering in Medicine; Biology Society (EMBC)*, 2021.

[13] Marcello Fusca, Francesco Negrini, Paolo Perego, Luciana Magoni, Franco Molteni, and Giuseppe Andreoni. Validation of a wearable imu system for gait analysis: Protocol and application to a new system. *Applied Sciences*, 2018.

[14] Nina Verdel, Miha Mohorčič, Miha Drobnič, Matej Supej, and Matjaž Depolli. Time synchronization in wireless imu sensors for accurate gait analysis during running. `https://event.unitn.it/ieee-star2023/`, 2023. Accessed: 2023-12-20.

[15] Matthew Rhudy. Time alignment techniques for experimental sensor data. *International Journal of Computer Science & Engineering Survey*, 5:1–14, APR 2014.

[16] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *IEEE Network*, 18(4):45–50, 2004.

[17] S. Johannessen. Time synchronization in a local area network. *IEEE Control Systems Magazine*, 24(2):61–69, 2004.

[18] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *IEEE Network*, 18(4):45–50, 2004.

[19] Ki Young Koo, David Hester, and Sehoon Kim. Time synchronization for wireless sensors using low-cost gps module and arduino. *Frontiers in Built Environment*, 2019.

[20] Nicholas Kostikis, George Rigas, Spyridon Konitsiotis, and Dimitrios I. Fotiadis. Configurable offline sensor placement identification for a medical device monitoring parkinson's disease. *Sensors*, 21(23):7801, November 2021.

[21] Bao Yang, Ying Li, Fei Wang, Stephanie Auyeung, Manyui Leung, Margaret Mak, and Xiaoming Tao. Intelligent wearable system with accurate detection of abnormal gait and timely cueing for mobility enhancement of people with parkinson's disease. *Wearable Technologies*, 3, 2022.

[22] Tommaso Proietti and Andrea Bandini. Wearable technologies for monitoring upper extremity functions during daily life in neurologically impaired individuals. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 32:2737–2748, 2024.

[23] Jian Li, Jian Peng, Zhen Lu, and Kemin Huang. The wearable lower limb rehabilitation exoskeleton kinematic analysis and simulation. *BioMed Research International*, 2022:1–10, August 2022.

[24] Timothy M. McGrath, Gordon Waddington, Jennie M. Scarvell, Nick B. Ball, Rob Creer, Kevin Woods, and Damian Smith. The effect of limb dominance on lower limb functional performance – a systematic review. *Journal of Sports Sciences*, 34(4):289–302, June 2015.

[25] Amy O. Parkinson, Charlotte L. Apps, John G. Morris, Cleveland T. Barnett, and Martin G. C. Lewis. The calculation, thresholds and reporting of inter-limb strength asymmetry: A systematic review. *Journal of Sports Science and Medicine*, page 594–617, August 2021.

[26] Chris Bishop, Anthony Turner, and Paul Read. Effects of inter-limb asymmetries on physical and sports performance: a systematic review. *Journal of Sports Sciences*, 36(10):1135–1144, August 2017.

[27] Joshua A. J. Keogh, Emma E. Waddington, Zaryan Masood, Sobia Mahmood, Anil C. Palanisamy, Matthew C. Ruder, Sameena Karsan, Chris Bishop, Matthew J. Jordan, Jennifer J. Heisz, and Dylan Kobsar. Monitoring lower limb biomechanical asymmetry and psychological measures in athletic populations—a scoping review. *Scandinavian Journal of Medicine & Science in Sports*, 33(11):2125–2148, August 2023.

[28] A. Vilhar and M. Depolli. Improving the dynamics of off-line time synchronization in wireless sensor networks. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0285–0289, 2018.

[29] Andrej Vilhar and Matjaž Depolli. Time synchronization problem in a multiple wireless ECG sensor measurement. In *14th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pages 83–86, 2018.

[30] Aleksandra Rashkovska, Matjaž Depolli, Ivan Tomašić, Viktor Avbelj, and Roman Trobec. Medical-grade ECG sensor for long-term monitoring. *Sensors (Basel, Switzerland)*, 20(6), 2020.

[31] Lee J. Bain. Applied regression analysis. *Technometrics*, 9(1):182–183, 1967.

[32] Mathieu Lepot, Jean-Baptiste Aubin, and François H.L.R. Clemens. Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. *Water*, 9(10), 2017.

[33] David Middleton. Statistical theory of signal detection. *Transactions of the IRE Professional Group on Information Theory*, 3(3):26–51, 1954.
[34] N Weissman, A Katz, and Y Zigel. A new method for atrial electrical activity analysis from surface ecg signals using an energy ratio measure. In *2009 36th Annual Computers in Cardiology Conference (CinC)*, pages 573–576, 2009.
[35] Robert G Brown. Exponential smoothing for predicting demand. *Operations Research*, 5(1):145–145, 1957.

**Matjaž Depolli** received his PhD in computer and information science from Jožef Stefan International Postgraduate School Ljubljana in 2010. He currently holds the position of senior research fellow at the Department of Communication Systems at the Jožef Stefan Institute in Ljubljana.

**Nina Verdel** received her PhD from the Faculty of Mathematics and Physics at the University of Ljubljana in 2020. She currently works as an assistant professor at the Faculty of Physical Education at the University of Ljubljana and as a PhD assistant at the 'Jozef Stefan' Institute.

**Gregor Kosec,** head of the Parallel and Distributed Systems Laboratory, obtained a Ph.D. in 2011 at University of Nova Gorica. In the same year he became a member of Parallel and Distributed Systems Laboratory at Jožef Stefan Institute.