electronics

MDPI

Article

ecoSync: An Energy-Efficient Clock Discipline Data Synchronization in Wi-Fi IoMT Systems

Steven Puckett [1,*] and Emil Jovanov [2]

[1] Sanders College of Business, University of North Alabama, Florence, AL 35632, USA
[2] College of Engineering, University of Alabama at Huntsville, Huntsville, AL 35899, USA; jovanoe@uah.edu
* Correspondence: spuckett1@una.edu; Tel.: +1-205-994-4130

Abstract: The growth of the Internet of Medical Things (IoMT) and healthcare data analytics allows wearable Wireless Body Area Networks (WBANs) and ambient sensors to collect the large quantities of physiological signals necessary for better patient diagnostics and treatments. Artificial intelligence and machine learning algorithms frequently require precisely synchronized signals from multiple sensors, which in turn require time-consuming and energy-inefficient synchronization methods with constant wireless network connectivity. We propose ecoSync, a highly energy-efficient time synchronization algorithm for Wi-Fi devices in IoMT applications. We demonstrated that ecoSync can correct the time difference error to ±42 µs with an hour between resynchronizations, using only 658 millijoules of energy. This is an 87% improvement in time difference error and a 99.93% reduction in energy usage over using TSF for synchronization alone over a 1 h period. Wireless synchronization of sensors allows placement of physiological sensors on objects of everyday use (Smart Stuff), which in turn allows seamless collection of physiological status data every time we interact with smart objects in an IoMT environment.

Keywords: IoMT; synchronization; Wi-Fi; TSF; PTP; clock relation model; clock discipline algorithm

Citation: Puckett, S.; Jovanov, E. ecoSync: An Energy-Efficient Clock Discipline Data Synchronization in Wi-Fi IoMT Systems. *Electronics* **2023**, *12*, 4226. https://doi.org/10.3390/electronics12204226

Academic Editors: Paul Kwan, Tony Jan and Rajan Kadel

Received: 3 September 2023
Revised: 9 October 2023
Accepted: 10 October 2023
Published: 12 October 2023

1. Introduction

The advent of new healthcare sensors and their integration using the Internet of Medical Things (IoMT) and Wireless Body Area Networks (WBANs) have started to generate a massive growth in healthcare patient data. New machine learning (ML) and artificial intelligence (AI) algorithms process these data streams in near-real time in the cloud. Many IoMT sensors utilize Wi-Fi to send data to a cloud service for processing and storage due to the low cost and functionality of embedded, system-on-chip (SoC) platforms like the Espressif ESP32 family of microprocessors [1]. These platforms are highly energy efficient in deep sleep modes, possess built-in Wi-Fi/Bluetooth, and are well-suited for IoMT system integration. However, as with all wireless systems, power utilization increases significantly during transmissions, reducing the battery lifetime if the device is left connected to Wi-Fi for long periods, significantly influencing user factors such as battery life and sensor weight.

Due to the low cost of IoMT platforms, health sensors can be developed for highly specific data collection, such as electrocardiography (ECG), photoplethysmography (PPG), or inertial sensors (activity monitors). These individual health sensors connect to Wi-Fi and send their data independently of other devices used by the patient. The lack of data synchronization between individual transmissions represents a massive issue for ML and analysis. For example, suppose a patient wears an ECG and a PPG monitor that communicate through a WBAN with the personal or home server to assess blood pressure using pulse travel time assessment (latency of PPG and ECG) [2]. Precise time synchronization is essential for this application for real-time assessment and monitoring. Having the data synchronized to a common clock allows the data to be directly processed by ML models and other algorithms regardless of the platform, communication technology, or equipment manufacturer.

Energy efficiency is a key part of the IoMT and BAN architecture since it determines the total lifetime for a given battery, as well as the size and weight of the battery, as one of the critical user factors influencing sensor technology acceptance. However, research projects frequently overlook energy efficiency when developing new Wi-Fi time and clock synchronization systems. Most of the proposed systems require constant Wi-Fi connectivity or resynchronization of the clocks over very short intervals [3–8]. Thus, battery life is sacrificed for clock synchronization. It has been shown that power efficiency and data reliability are both improved by offline monitoring with periodic data uploads [9]. The ability of IoMT devices to timestamp data while being offline for long periods to conserve power is of great importance for remote and wireless monitoring of patients.

For healthcare providers such as hospitals, nursing homes, and assisted living facilities, data synchronization also becomes an issue of location. For these types of environments, there are multiple Wi-Fi Access Points that the sensors could connect to depending on patient movement. It would also be beneficial to have fixed monitoring equipment, such as treadmills used in stress tests or rehabilitation, synchronized to the same clock source as those used by the IoMT devices.

Monitoring of stress and precise diagnosis of cardiovascular issues would greatly benefit from unobtrusive monitoring during activities of daily living. To remove the need for cables, which would interfere with activities of daily living, each of the sensors is implemented as a standalone wireless sensor. Traditionally, the monitoring system would collect heart rate and heart rate variability to assess Autonomous Nervous System (ANS) and stress-induced changes. If the system allows synchronized monitoring of multiple sensors, which was our primary objective, we can assess Pulse Wave Velocity (PWV) as the speed of blood pulse after each heartbeat, and Pulse Arrival Time (PAT)/Pulse Travel Time (PTT) as a latency between electrical activation of the heart muscle (R peak of ECG) and pulse arrival at a certain point on body measured by PPG [10,11]. PWV can be calculated as PTT between sensors divided by the distance between two PPG monitoring locations. Multiple locations could be used to assess PAT/PTT using PPG, such as the wrist, the ear, or the foot. Moreover, synchronized sensors allow the placement of physiological sensors on objects of everyday use that we call Smart Stuff, such as a smart water bottle [12]. Therefore, we might collect a snapshot of ANS activity every time we interact with smart objects in the IoMT environment.

A comparison of several synchronization protocols that could be used on IoMT devices to determine their energy efficiency, synchronization performance, and usability for healthcare campus environments is evaluated along with our proposed solution. We propose ecoSync, a highly energy-efficient clock synchronization utilizing a combination of Precision Time Protocol [13], Wi-Fi's Time Synchronization Function (TSF) [14], clock relation models (CRM), and clock discipline algorithms (CDA) to provide sub-millisecond synchronization as appropriate for use in healthcare environments.

This paper is organized as follows. Section 2 provides an overview of the current clock and time synchronization protocols and clock relation models evaluated within this paper. Section 3 discusses related works on data and clock synchronization over Wi-Fi. Section 4 describes the proposed architecture and protocols for ecoSync, an energy-efficient healthcare campus-wide data synchronization deployment. Section 5 presents the results and comparison to existing time synchronization protocols. Section 6 concludes this work.

2. Timestamp and Synchronization Protocols

Several clock synchronization protocols are currently used for both wired and wireless networks. For wireless networks, most protocols require the access point, gateway, or cellular tower to broadcast a beacon at set intervals with timestamps that the end device can use to sync its internal clocks. This requires that the end devices are always awake and listening for these beacon packets, or the devices must wake up at specified times to receive the packets before going back to sleep [15]. However, this is not an acceptable solution in IoMT and WBAN devices because of the limited power budget, and waking

up for every beacon is inefficient. If this process is not part of the protocol, the beacons could create network congestion, which increases retransmission, thus further depleting power resources. In the next subsections, we describe the most common protocols used for clock and time synchronization and provide an energy usage analysis for comparison. The Joulescope JS220 was used to generate all power measurements [16] with the software running on an Espressif DOIT ESP32 DEVKIT V1, a low-cost SoC with built-in Wi-Fi. For all tests, the devices were initially connected to the AP; then, Wi-Fi was disabled on the device to emulate typical WBAN sensors. After 5 s, Wi-Fi was turned back on, and the device synchronized using the programmed protocol. The device then went back to sleep for another 5 s to repeat the cycle. Using a 5 s window provided a "hot" connection to the AP for faster association. If the measurements were made from a cold start, the power usage would be much higher due to the additional time reassociation and reconnection to the AP takes.

2.1. Network Time Protocol (NTP)

NTP (Network Time Protocol) is one of the first time synchronization protocols that is still widely used today. It is part of the TCP/IP suite and is initiated by an end device that initiates a time-request exchange with an NTP server. It takes approximately six exchanges over several minutes to initially set the clock, updates approximately every 17 min, and typically has a ±1 s accuracy. NTP is primarily used to set the date and time of an end device. The NTP packets are around 76 bytes per message [17] and can provide sub-seconds of accuracy with high resynchronization rates. It is not a good fit for wireless health monitoring devices due to its energy inefficiency and lack of clock synchronization accuracy. Our tests showed that the NTP uses an average of 376.5 millijoules [mJ] of energy for each synchronization event, as seen in Figure 1.

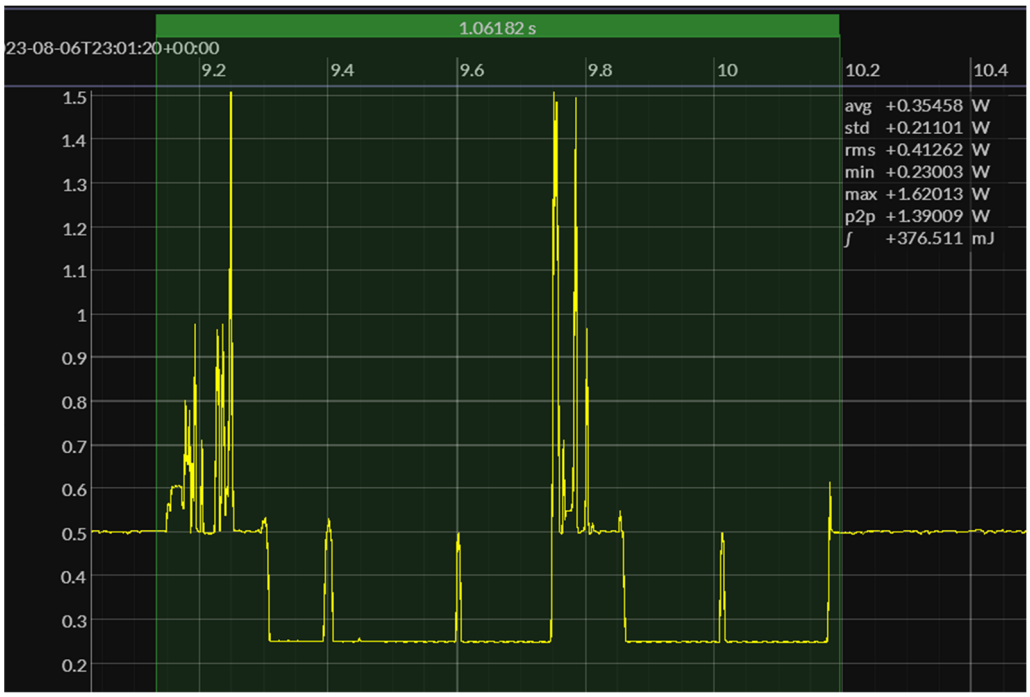

Figure 1. NTP energy usage during resynchronization (mW).

2.2. Simple Network Time Protocol (SNTP)

Simple Network Time Protocol [18] is a simplified version of NTP that is far less secure than NTP. It is typically used on microcontrollers or other hardware in which accurate clock or time synchronization is not required and is also used to set on-board real-time clock (RTC) date and time. The SNTP's accuracy is limited to fractions of seconds, multiple

seconds, or minutes, depending on how it is deployed. Figure 2 shows that the ESP32 utilizes 425 millijoules of energy for each SNTP resynchronization.

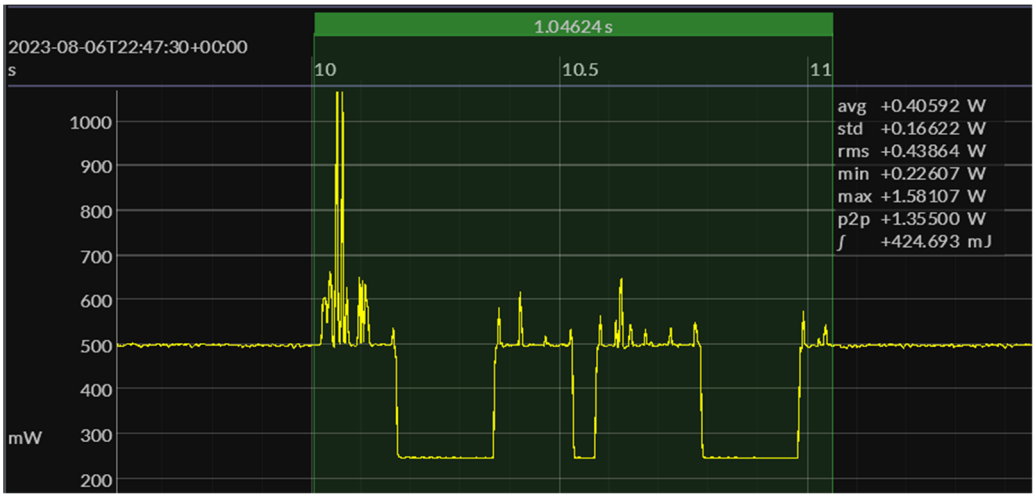

Figure 2. SNTP energy usage during resynchronization.

2.3. Precision Time Protocol (PTP)

IEEE 1588v2 Precision Time Protocol (PTP) [19] was developed for Ethernet-based networks to provide highly precise frequency and time clock synchronization to ensure base station stability and handovers. Slave clocks are synchronized to a Grandmaster clock with sub-nanosecond granularity without the need for any administration. The devices are kept in sync with the transmission of timestamps that are sent within PTP packets. PTP calculates the propagation delay over the Ethernet network using a delay request-response mechanism, as seen in Figure 3.

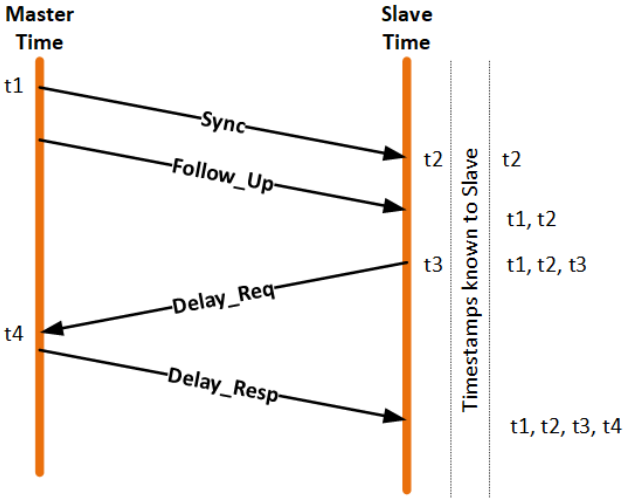

Figure 3. IEEE 1588 PTP Message Exchange.

This delay is then used to determine the master timestamp, which is sent using the Sync and Follow_Up messages. The slave device then sends the Delay_Req to the master, and it responds with the Delay_Resp message. The slave marks the timing of each of these four messages and can then calculate the mean propagation delay (*tmpd*) of the message path (Equation (1)).

$$tmpd = \frac{(t2 - t1) + (t4 - t3)}{2} \quad (1)$$

The slave uses the mean propagation delay, the *Sync* message, and the Follow_Up message times to calculate the clock offset of the slave from the Grandmaster clock (Equation (2)).

$$Clock\ Offset = t2 - t1 - tmpd \quad (2)$$

The PTP packet header is 34 bytes in length and is common for all messages. The Sync, Follow_Up, and Delay_Req messages all add 10 bytes to the header (44 bytes in total), and the Delay_Resp message adds 20 bytes to the header (54 bytes in total). Unlike the NTP, PTP only requires four messages for highly accurate clock synchronization with almost half the packet size.

The Delay_Req message is the same as the *Sync* message, but it is instead sent by the slave node to the master clock. It includes the time the message was sent with respect to the slave clock's time. The master device then sends a Delay_Resp message with the time the Delay_Req was received by the master. The delay request response message can be sent from the slave in order to resync its internal clocks as needed. The requestingPortIdentity field identifies the slave that originated the request.

With PTP designed to have clock synchronization accuracies in the sub-microsecond range, this makes the IEEE 1588 specification a good candidate for creating highly accurate synchronized timestamps in a healthcare campus. In our proposed solution, we utilized PTP to synchronize Wi-Fi access point CPU clocks to a central clock. The PTP is designed for continuous clock resynchronization to keep microsecond accuracies. On the ESP32, this means that a continuous Wi-Fi connection is required. A 5 s window was selected to measure the energy utilization of the PTP. The 5 s interval provides enough time for the Wi-Fi connection and for the PTP synchronization to occur. It also represents the same time window used to compare energy usage with other protocols. During the 5 s window, including connecting to the Wi-Fi network, PTP utilizes 1.93 joules of energy due to the protocol's cyclic synchronization messages, as seen in Figure 4.

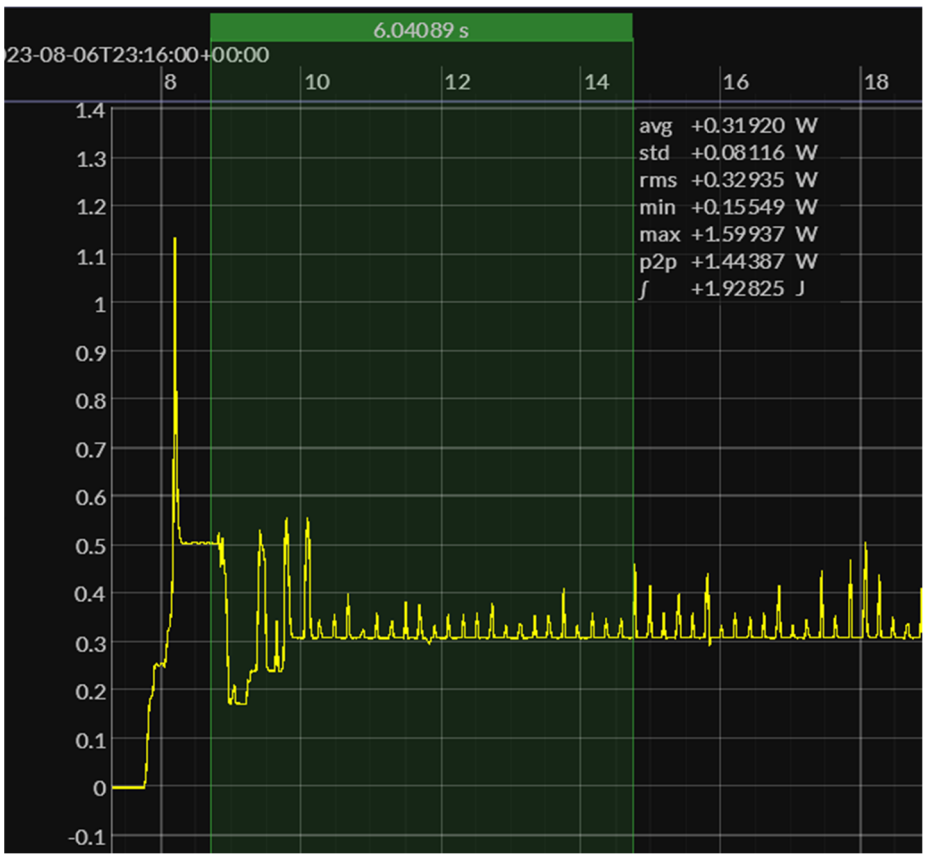

Figure 4. PTP energy usage (mW) during Wi-Fi connection; continuous synchronization for 5 s.

2.4. Wi-Fi Time Synchronization Function (TSF)

The Wi-Fi Time Synchronization Function (TSF), as described by the 802.11 standards [20], outlines synchronization between Access Points (APs) and wireless clients using 64-bit timers using 1 MHz clocks. These clocks are part of the Wi-Fi chipsets on the devices, and the timers are set to increment at 1 µs intervals. The client clock timer is required to be accurate to within ±100 ppm by the standard.

The AP sends its TSF timer value inside beacon frames every 102.4 milliseconds to the clients, who then overwrite their TSF timer with the AP TSF value within the frame (see Figure 5). This synchronization is primarily used to coordinate access to wireless channels and is not typically utilized for other functions. TSF does not account for propagation delay between the AP and the clients. Propagation delays can introduce delay and jitter depending on the range, retransmission rates, network congestion, and power savings settings of the clients.

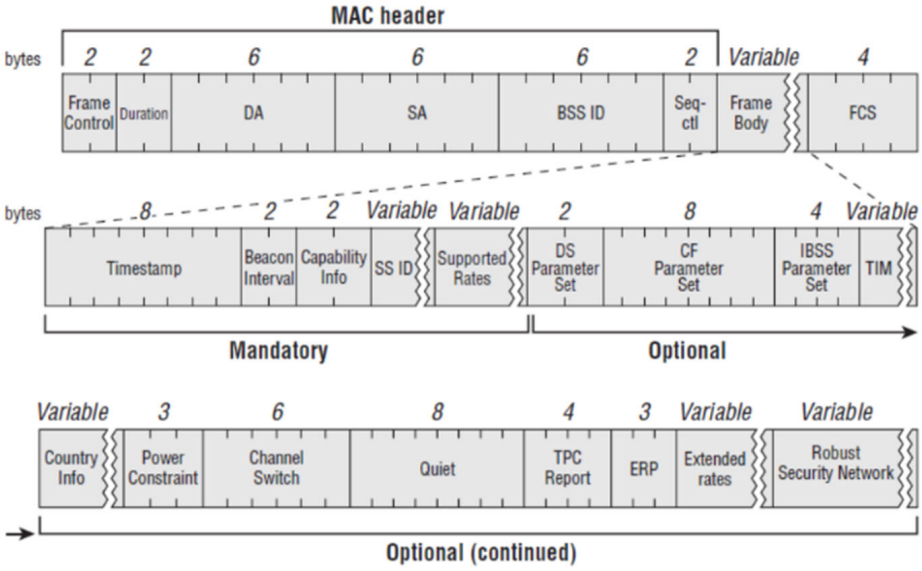

Figure 5. IEEE 802.11 Wi-Fi Beacon Frame.

In the IEEE 802.11-2012 standard, the Timing Advertisement (TA) method was introduced. This addition allows an AP to send UTC time and TSF using the beacon frame. The beacon contains the TSF timer value and the offset between the local AP clock and the TSF timer. The client can then use the received TSF value and offset to resynchronize its own internal clocks. One problem with this layout is that the AP and client local clocks are typically running at different rates, which affects the offset. Another is that jitter is introduced since software at higher levels in the client is needed to set the local clocks [21,22]. The IEEE 802.11-2012 does not identify how the AP or the clients are to utilize the TA method for synchronization, and so this functionality is not currently included in most IoMT devices.

Since the TSF information is part of the standard Wi-Fi protocol, there is very little overhead when synchronizing the TSF clock with the ESP32 master clock. The TSF value is automatically transmitted to the ESP32 anytime it is connected to the AP. The only additional energy required is to keep the ESP32′s master clock powered on and any clock relation models and clock discipline algorithms for keeping the master clock in sync with the TSF clock. Figure 6 shows the TSF delay error between the AP and the client. Figure 7 shows that TSF synchronization uses 1.4 joules of energy over a 5 s cycle.

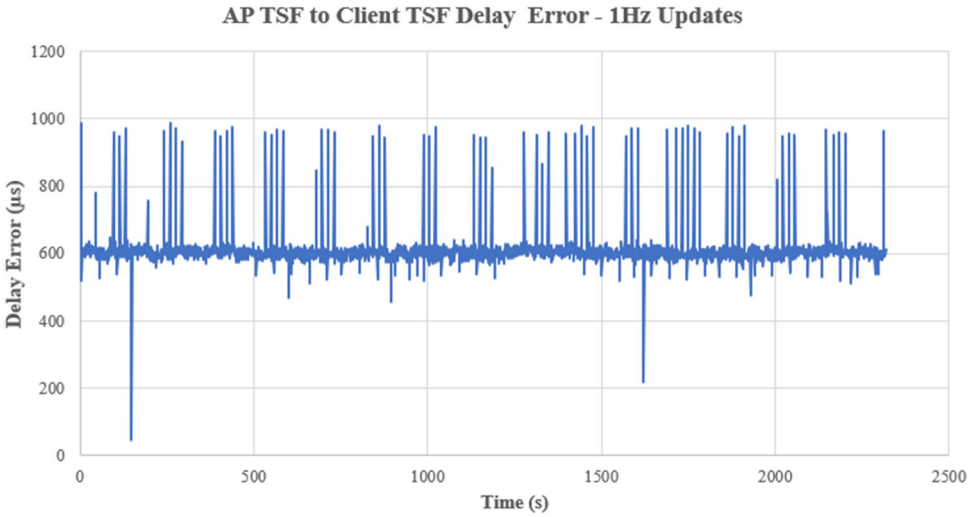

Figure 6. TSF AP to client delay error at 1 Hz resynchronization.

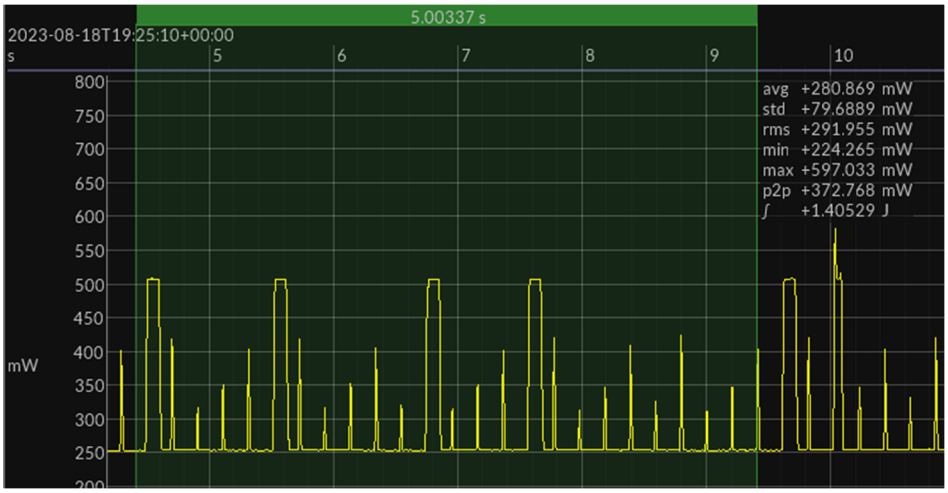

Figure 7. TSF energy usage over a 5 s interval.

2.5. Clock Relation Models (CRM)

A clock is a counter that measures the number of precise ticks from an oscillator in reference to some master time or an epoch. A perfect clock would have no drift or skew and would not be affected by environmental or manufacturing conditions. A perfect clock would give $C_{perfect}(t) = t$. However, this is not the case in the real world, where clocks, especially those on low-cost microprocessors, can be significantly impacted by these conditions, which cause errors of several hundred parts per million (ppm). D.W. Allan [23] developed the time deviation of a continuous clock in Equation (3). This equation includes frequency drift ω, frequency offset γ, time offset θ, and random errors ε at an instant t.

$$C(t) = t + \theta + \gamma t + \omega t^2 + \epsilon(t) \tag{3}$$

For precise clock synchronization, a clock relation model can utilize Equation (3) to model some clock, $C_2(t)$, to a reference (master) clock, $C_1(t)$, at some time t. Yiğitler et al. provide a detailed mathematical proof of the clock-to-clock relation model based on Equation (3), which gives a more detailed clock relation model in Equation (4) [24].

$$C_2(t) = \alpha_{12} C_1(t) + \theta_{12} + \varepsilon_{12}(t) + (1 + \gamma_2) d_{12}(t) \tag{4}$$

Within Equation (4), α_{12} is the clock skew between the clocks and $d_{12}(t)$ is the message delay that consists of both a constant delay and a random, variable delay that is a function of t. The frequency drift, ω, can be dropped from the equation if the time synchronization algorithm resyncs the clocks with less than 10 min intervals [25].

2.6. Clock Discipline Algorithms (CDA)

The primary purpose of the clock discipline algorithm (CDA) is to estimate the CRM parameters, then calculate the reference clock value using the CRM equation. This allows the local clock to be synchronized or adjusted to the master clock with a lower error threshold and a higher accuracy. CDAs can also be utilized to adjust timestamps for data measurements in relation to both clocks for further processing by other data software programs. A few of the clock discipline algorithms are evaluated and described below.

Offset-Only: The only parameter that the algorithm must determine is the time offset between the clocks, τ_{12}, as shown in Equation (5). The clock skew, α_{12}, is considered a constant and equal to 1. This algorithm is efficient but requires constant resynchronization of the clocks to keep the estimation error low. During our testing, it was found that the clock skew between the CPU clock and the TSF clock was too high for an offset-only model.

$$\tau_{12}(t) = C_2(t) - C_1(t) \tag{5}$$

Linear Estimator: This algorithm treats the clock synchronization as a linear relationship if the frequency drift is ignored. The clock relation model shown in Equation (4), can be written as a linear equation in the form of $y = mx + b$ where m is the clock skew, α_{12}, and b is the sum of the estimated delay, offset, and random errors, as shown in Equation (6). This is the model that is used for ecoSync.

$$C_2(t) = \alpha_{12}C_1(t) + \theta_{12} + \varepsilon_{12} + d_{12} \tag{6}$$

3. Related Work

There has been much research into utilizing PTP and TSF for Wi-Fi client synchronization. Chen and Yang [6] and Aslam et al. [26] took an in-depth look at utilizing software and hardware-based PTP over Wi-Fi networks, specifically using the TSF timer as the reference clock. Likewise, Mahmood et al. [21] conducted a survey of PTP, TSF, and other methodologies for clock synchronization over Wi-Fi and determined that oscillator wander is the dominant noise source for low synch rates of 10 s or more. Coviello et al. [3,27] found that for data synchronization, accuracy between clocks is more important than accuracy to true UTC time, and that data can be synchronized across multiple nodes as long as they are accurate to a master clock. However, in all cases, the synchronization algorithms proposed still require constant wireless connectivity, thus being very energy inefficient. The concept of Wireless Time-Sensitive Networking (TSN) was investigated, and it was found that many of the wireless time synchronization protocols were adequate, but all cases required constant resynchronization [28] and were not energy efficient. However, Coviello et al. found that clock synchronization and power savings were possible by calculating the number of skipped synchronizations to keep the error within a threshold [29]. Time-synchronization for other types of Master–Slave WBANs utilizing FLSA to sync the master to all slaves is introduced, but does not answer the problem of high-energy utilization for synchronization [30].

Haxhibeqiri et al. [5] deployed PTP over Wi-Fi to determine how the synchronization error would be impacted by modifying the beacon interval from 50 ms to 1000 ms. They found that beacon intervals under one second had little impact on the overall synchronization accuracy. They also utilized PTP to sync between APs and found that nodes connected to them could be synchronized to $\pm$16 µs if both APs are synched appropriately with little network congestion. Romanov et al. [4] used TSF from the 802.11 beacon frame and utilized an FPGA that the wireless network interface card (WNIC) triggers to read and resync the timers. Though this process showed that sync errors and jitter were greatly reduced, it

requires specialized hardware, extremely small resync intervals, and constant power to the WNIC while connected to the AP. Avitabile et al. utilized a slave timer correction algorithm based on the number of synchronization slots skipped in which the radio was turned off between synchronization time slots to save power [8,9]. This was one of the few studies looking at synchronization and power savings.

A majority of the papers focused on creating the lowest synchronization time and error rates for wireless clients that are always connected to the AP. Nearly all the papers kept the Wi-Fi connection to the clients on and had resynchronization rates between one millisecond and one second, with little to no emphasis on power savings or long-term sleep cycles often used in battery-powered sensor nodes. Also, most of the articles did not combine CRM and CDA with these protocols in order to further reduce synchronization errors or save energy. Yiğitler et al. compared several CDA and CRM models with respect to wireless sensor networks and IoT devices [24] but did not evaluate their energy efficiency. Cappelle et al. were able to obtain a synchronization accuracy of 200 µs using an energy consumption of 198 mJh, but their solution worked only over Bluetooth Low-Energy (BLE) devices [31].

4. EcoSync

We propose energy-efficient clock optimization and synchronization (ecoSync) as an energy-efficient, sub-millisecond time synchronization algorithm for IoMT and WBAN devices. Most IoMT devices are battery operated, so energy efficiency is a major requirement. There are many clock relation models and algorithms that are highly accurate, but require constant synchronization. For our solution, we are only interested in algorithms that work with long resynchronization delays to reduce wireless connectivity, thus reducing energy consumption. This means that the algorithms need to estimate the time differences using few synchronization points that are very far apart in time. Another difference from the other solutions is that the accuracy for IoMT devices requires only sub-millisecond synchronization of data, not microseconds of accuracy which most researchers are trying to achieve. Heart monitors (i.e., ECG) usually sample signals at 100 to 500 Hz. Electroencephalography (EEG) uses sampling at most of 1 KHz, but not for wearable applications. Other types of sensors use a sampling frequency of 100 Hz or less. This provides some flexibility in the design of an IoMT clock CDA since the threshold we are designing for is less than 1 millisecond of error.

4.1. Healthcare Campus Clock Domains

Our proposed solution is to create a synchronized healthcare campus environment for IoMT devices that can work over a network of several Wi-Fi Access Points (AP). However, the TSF clock within each AP cannot be modified or reset outside of the 802.11 protocol, as most Wi-Fi chipsets do not allow access to the TSF clocks. Also, it was found that resetting the TSF clock to synchronize to an outside clock caused the Wi-Fi connections to drop and become unusable [14]. There is also no feasible way to synchronize the AP TSF clocks together without specialized hardware; only the Wi-Fi clients' TSF clocks can be synchronized, as a slave, to the associated Access Point TSF clock.

We can overcome this issue by breaking the healthcare campus environment into two distinct clock domains. The primary clock domain consists of the Server Master Clock and the CPU clocks of the Access Points. As discussed earlier, PTP can be deployed between the Access Points and the server acting as the Master Clock to provide a highly accurate, synchronized domain between the Access Points and the Server as seen in Figure 8. This requires that the software running on each AP needs to determine the offset between its TSF clock and the PTP-synchronized CPU clock for UTC time synchronization, if required. Since this has already been proven and well-studied, we do not cover it within this work [6].

The secondary clock domains consist of an Access Point's TSF clock, and the Wi-Fi clients' TSF and CPU clocks. The Access Point's TSF clock acts as the Master Clock for all Wi-Fi clients connected to it. Once the client has associated with an Access Point, the client begins to receive beacon packets that include the Access Point TSF clock value as a 64-bit

number. The client then resets its own TSF clock to this value to synchronize the two clocks together. If the device turns off its Wi-Fi radio or is no longer connected to the Access Point, the client TSF clock is shut down and no longer active. The client must re-establish its association with an Access Point before the TSF clock is resynchronized.

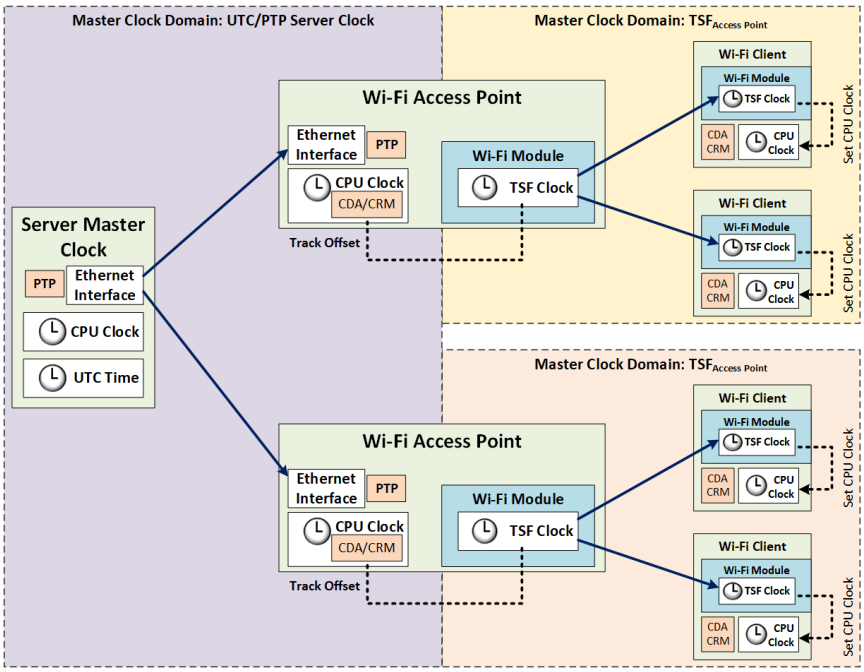

Figure 8. Proposed campus-wide clock synchronization domains.

4.2. IoMT Device Clock Discipline Algorithm

The ecoSync algorithm (Algorithm 1) is designed as a cyclic process within the embedded software loop using the clock relation model in Equation (4). The algorithm first reads the client TSF clock once a Wi-Fi connection is made, then syncs the client CPU clock with the TSF clock. This is a software function, so reading the TSF clock and resetting the CPU clock creates a jitter offset that can be measured internally; this offset is then added to the CPU clock. If this is the first synchronization due to a power cycle of the device, or if the device moves to a new Access Point, a timer is set for a short delay (10 s or less) to initiate a new TSF sync. This second measurement is then used with the initial reading to calculate clock skew, delay, offset, frequency drift, and any random errors for input into the CRM. The model can then estimate when the device should resynchronize given some error threshold. This provides a theoretical maximum time between resynchronizations to optimize the power savings of the device. An internal clock timer is then set to this time in which a new synchronization occurs. A comparison of the new TSF reading and the current CPU clock is then used to determine the accuracy of the CRM parameters, and adjustments are made as necessary. A new resynchronization time is set and the algorithm repeats. Utilizing the error correction at each synchronization cycle allows the algorithm to react to changes in the system that require changes to resynchronization windows. The error threshold can also be modified to meet the data synchronization timing requirements for each device.

In several use cases, the data collected by the device is cyclic at predetermined frequencies. One example is with heart monitoring, in which the ECG analog signal is read every 300 Hz, or 3.33 milliseconds. This information can be used with the CRM model as a means of error correcting collected data by backpropagation using the Least Squares Method or some other regression model. This method is also used to significantly increase the resynchronization window, even allowing the data collection time to be beyond the allowed error. The CRM backpropagation would recalculate each collected data timestamp for the past cycle to improve errors in the recorded time.

Algorithm 1. EcoSync

```
InitialDeviceSync() {
    Connect to Wi-Fi
    Read TSF_client Clock
    Set CPU Clock_client
    IF Initial Synchronization {
        while(! ResyncWindow_initial) {} //wait for sync window
        Read TSF_client Clock
        Read CPU Clock_client
        Calculate CRM attributes
        Calculate ResyncWindow_next
        }
 }

 LOOP {
    if(Clock_client > ResyncWindow_next) {
        Connect to Wi-Fi
        Read TSF_client Clock
        Read CPU Clock_client
        Calculate time difference error
            If(ResyncWindow_previous > WindowThresh) {
                Adjust CRM attributes
            }
        Calculate CRM attributes
        Calculate ResyncWindow_next
        Back-propagate recorded data timestamps using CRM
       Send data to server
       Turn-off Wi-Fi to preserve power
  }
}
```

5. Performance Evaluation

We evaluated the performance of ecoSync using a WBAN with two sensors and a gateway, as seen in Figure 9. Both sensors and the gateway sample used the same test signal to evaluate the signal detection latency caused by clock desynchronization. The evaluation setup utilized the Espressif ESP32 DEVKIT V1 as sensors (Wi-Fi clients) connected to a Raspberry Pi configured as a gateway (wireless Access Point). An Analog Devices ADALM2000 was configured as a signal generator with a 1 Hz square wave at 10% duty cycle providing a 100 millisecond pulse every second. A GPIO pin was configured on each ESP32 and the Raspberry Pi to trigger an interrupt when the pulse was detected. The interrupt calls a function that reads the CPU clock as a 64-bit integer in microseconds and writes it to an attached SD card along with a counter value and the ecoSync corrected value. When the CPU clock reaches the resync window value, the client reads the TSF clock and writes this value to the SD card along with the counter and new CPU clock value. This data can then be used to determine the time difference error between the TSF/CPU clocks, and the client-to-client time difference errors. The data from each device are read into a spreadsheet and the sample counts that each device recorded at each signal trigger are synchronized. The time difference in microseconds is then evaluated between the client nodes, and each node to the AP. This provides the node-to-node clock difference, and the node–AP clock difference for each device. As the sample numbers increase each second, the clock skew also increases. The actual clock skew can then be determined and compared with the ecoSync corrected times.

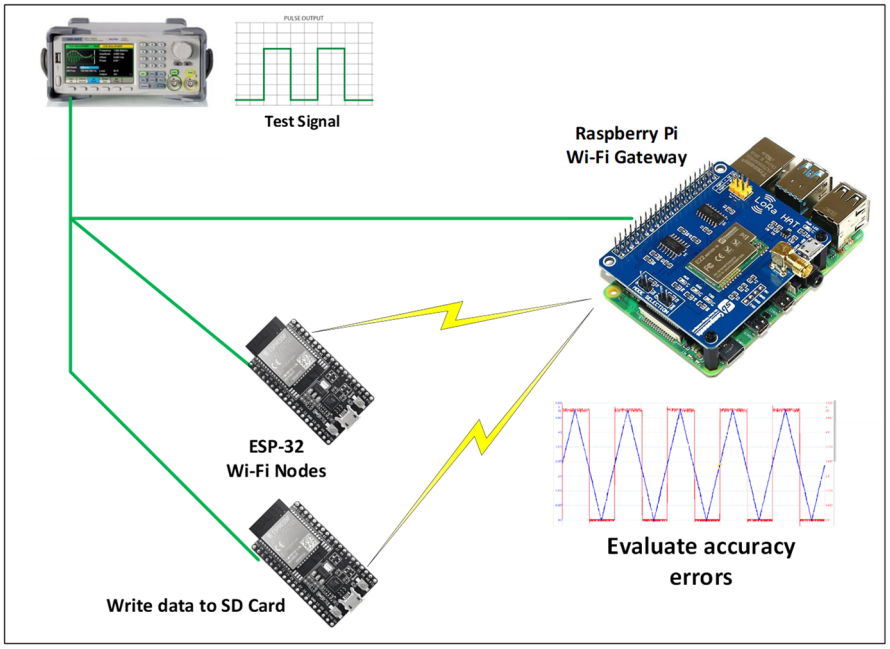

Figure 9. Experimental setup.

5.1. Time Synchronization Protocol Comparison

Time synchronization of sensors in the experimental setup over a one-hour period is shown in Table 1. Both NTP and SNTP are designed to be accurate to within ±1 s and require resynchronization within a few minutes to prevent errors greater than 1 s. For our measurements, we utilized a 5 min window with the ESP32 turning off Wi-Fi to conserve power between synchronizations. It was found that the clients stayed within ±1 s for the entire 1 h test using NTP and SNTP. This accuracy is not accurate enough for the WBAN synchronization necessary for IoMT devices. In contrast, ecoSync was found to be accurate to within tens of microseconds of accuracy.

Table 1. Time sync protocol comparison over 1 h.

Protocol	Time Format	Client-to-Client Error	Resync Time	Energy (J)
NTP	Epoch time (s)	±1 s	5 min	4.518
SNTP	Epoch time (s)	±1 s	5 min	5.106
PTP	64-bit integer (µs)	±15 µs [14]	Continuous	1389.6
TSF	64-bit integer (µs)	± 340 µs	Continuous	1011.8
EcoSync	64-bit integer (µs)	± 42 µs	60 min	0.6582

For both PTP and TSF synchronization, Wi-Fi was required to be active with the ESP32 connected to the Access Point in order to receive the PTP packets or the TSF beacons. This meant that the power utilization for the hour test was extremely high. As seen below, PTP uses significantly more energy due to the sync/delay messages requiring frequent updates.

In contrast, ecoSync relies on CDA and CRM models to accurately calculate the clock skew, offsets, and delays while the client is not connected to Wi-Fi. EcoSync showed an 87% improvement in client-to-client time difference error and a 99.93% reduction in energy usage over using TSF for synchronization alone.

5.2. EcoSync Evaluation

The ecoSync backpropagation algorithm was evaluated to determine the energy usage and cycle time needed for 3600 datapoints, assuming a 1 Hz sensor read cycle for one hour. It was found that using the clock relation model to cycle through the datapoints

before transmission only required 3.04 millijoules of energy and around 18 milliseconds to complete, as seen in Figure 10.

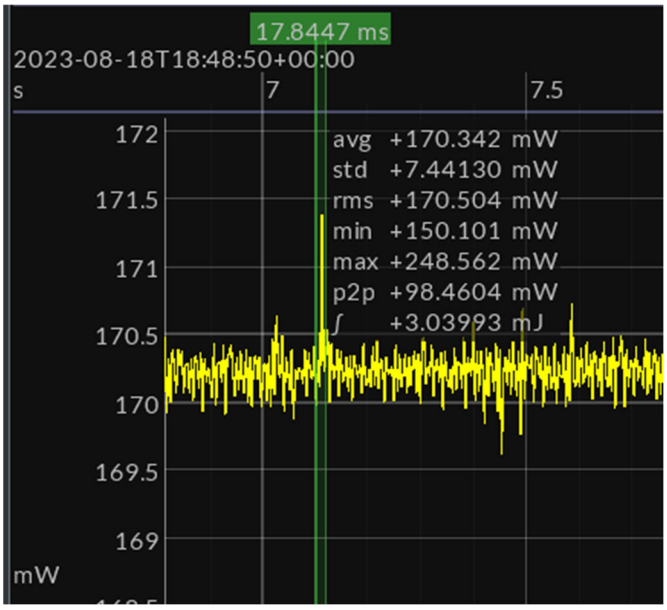

Figure 10. EcoSync clock discipline algorithm energy usage for 3600 datapoints (1 h of data at 1 Hz).

Figure 11 shows the full ecoSync cycle including Wi-Fi connection, CRM attribute calculations, and data backpropagation for the 3600 datapoints. The cycle required 658 millijoules of energy.

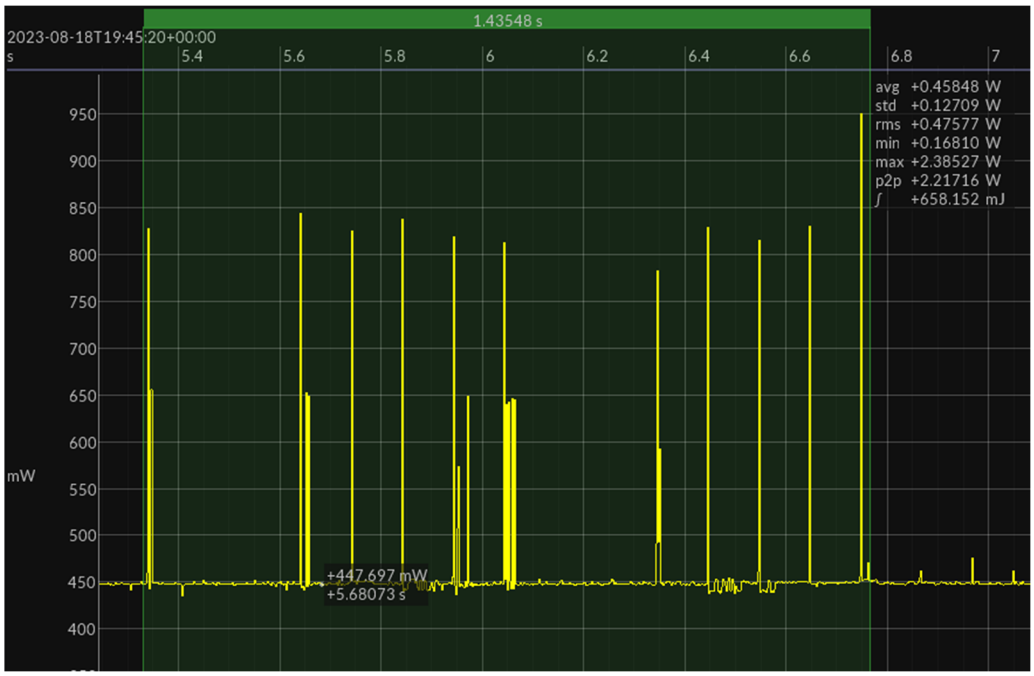

Figure 11. EcoSync with Wi-Fi connect/disconnect and data backpropagation.

5.3. Access Point to Client TSF Evaluation

The Access Point TSF clock and the client TSF clock time differences were evaluated to identify any delays and offsets that needed to be added to the clock relation model. It was found that an offset could be identified due to software, hardware, and transmission delays. This offset can be measured and accounted for within the clock relation model. For our

experiment, the offset delay was around 600 microseconds on average. Simply subtracting this offset reduced the error due to delays significantly, as seen in Figure 12.

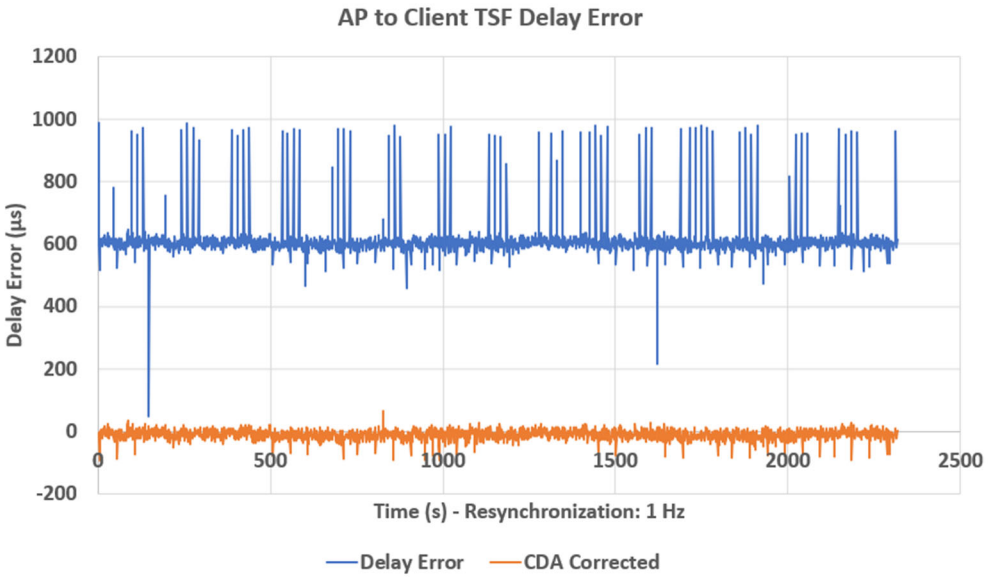

Figure 12. Access Point to client TSF synchronization delay error.

The spikes in the Delay Error are due to a known issue with the ESP32 TSF read function in which large delays in reading the register can occur. This is removed by measuring the time elapsed to read the values and removing this software jitter delay.

5.4. Client TSF Clock to CPU Clock Evaluation

The clock skew between the client's CPU clock and TSF clock was measured and found to be linear. This verified that finding the skew slope anywhere along this line would result in the ability to greatly reduce the time difference error between the two clocks. It also means that a single resynchronization time over a greater time period would also provide accurate clock relation model parameters. Figure 13 shows the comparison of the time difference error with the corrected clock time comparison over a 16 h period using only the calculated linear model.

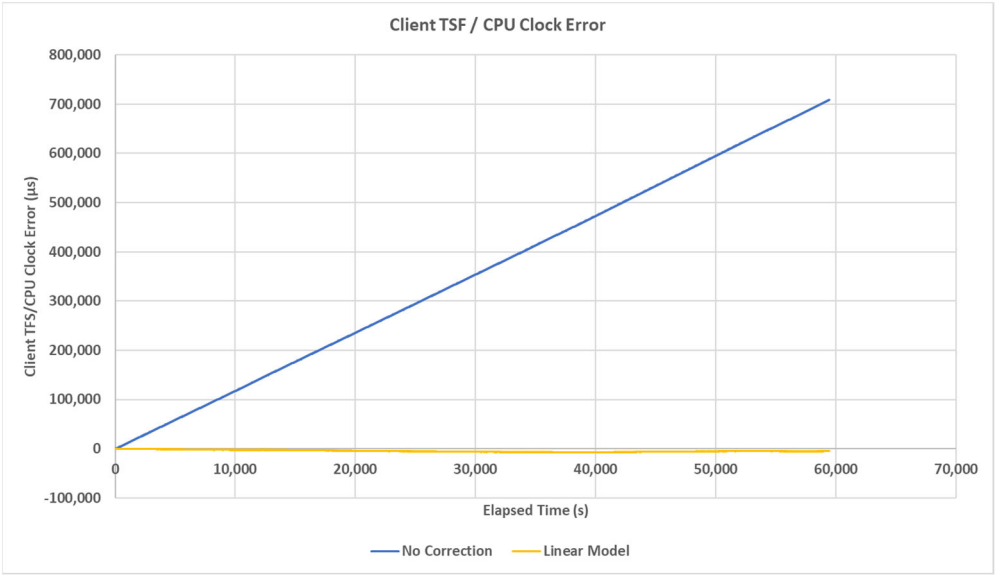

Figure 13. Client TSF/CPU clock error comparing no correction and CRM with offset.

5.5. Client TSF Clock to Client TSF Clock Evaluation

Measurements of the client-to-client TSF clocks determined that the time differences between the client TSF clocks were well below 100 micro-seconds, except for the occasional spikes as seen in Figure 14. This lines up well with the 802.11 synchronization error specification requirements for the TSF functionality and shows that utilizing the Access Point TSF as the Master Clock for the Access Point time domains is the right choice.

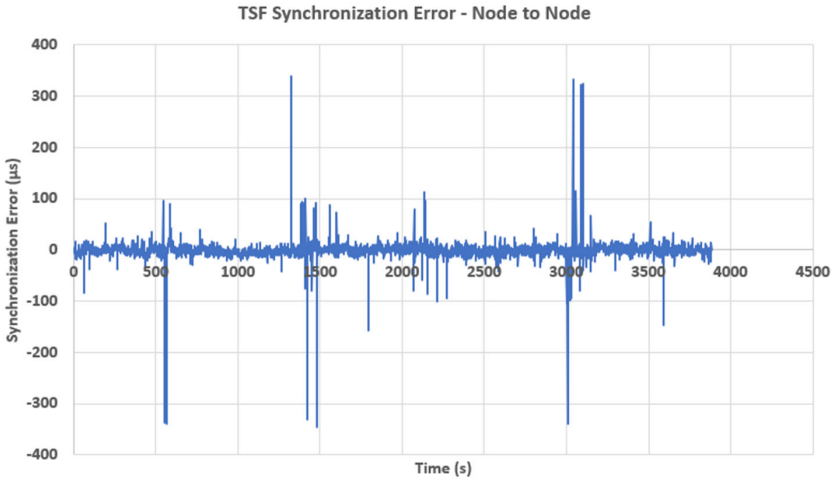

Figure 14. TSF clock error difference between clients.

5.6. Client CPU Clock to Client CPU Clock Evaluation

EcoSync requires a clock relation model for each client to synchronize its CPU clock with its TSF clock as well as the Access Point TSF clock. This requires the algorithm to be able to accurately determine the clock skews, offsets, software jitter, and delay errors for the CRM. Figure 15 shows the time difference errors after only 5 min. As you can see, without ecoSync, the client CPU clocks quickly diverge to greater than 6500 microseconds of error. The on-client TSF and CPU clock skew difference also increases significantly. With ecoSync, the client-to-client CPU clock difference stays well under 40 microseconds of error. Figure 16 shows the time difference error, using ecoSync, between two client CPU clocks with the initial synchronization and another after 1 h. The results show that utilizing the TSF clocks with clock discipline algorithms and clock relation models can synchronize data across IoMT devices to within tens of microseconds of accuracy with resynchronization times of greater than one hour.

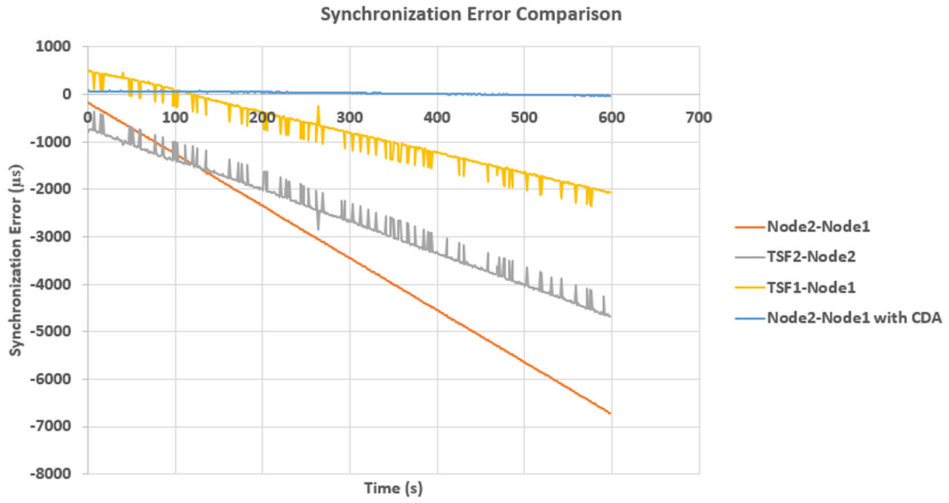

Figure 15. Client-to-client synchronization error.

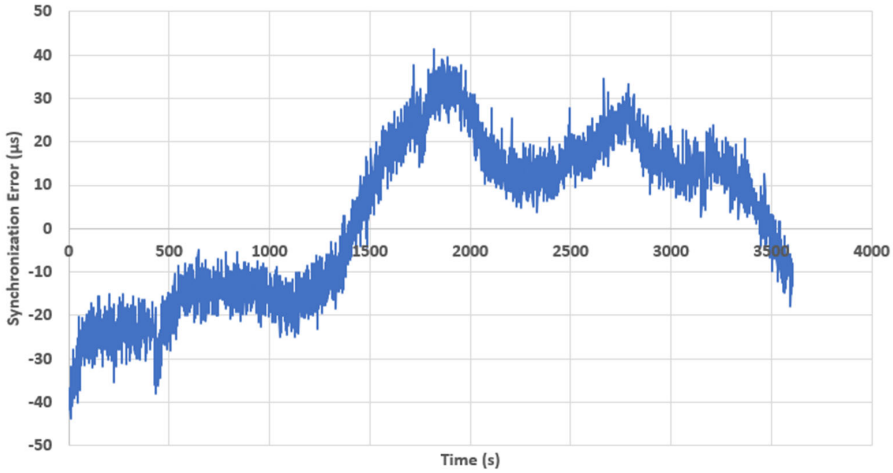

Figure 16. EcoSync client-to-client synchronization error.

5.7. Use Case: Calculating PTT Using Synchronized PPG and ECG Signals

To demonstrate the feasibility of the ecoSync time synchronization methods we implemented a system to measure PTT and PWV using an ECG sensor and two PPG sensors. The ECG sensor was based on the Analog Devices AD8232 single-lead ECG SoC [32] as built into the SparkFun AD8232 Single Lead Heart Monitor [33]. The Maxim MAX30102 [34] pulse oximeter and heart rate sensor were used for the PPG signals. The PPG sensors utilize the I2C communications protocol to connect to a microcontroller, which for our experiments was an ESP32 DEVKIT V1-DOIT with built-in 2.4 GHz Wi-Fi. The AD8232 uses an analog output from 0 V to 3.3 V for the ECG signal and was connected to the SEEED Studios XAIO ESP32C3, with 2.4 GHz Wi-Fi and a 12-bit ADC [35].

For both the PPG and ECG sensors, the microcontrollers were programmed to measure the ECG and PPG at 200 Hz. The signals utilized a baseline removal algorithm and all signals were passed through a software low-pass filter to remove power line interference at 60 Hz. The PubSubClient Arduino software library [36] was utilized without user authentication to send the messages using MQTT to a Raspberry Pi setup as a Wi-Fi Access Point (gateway). The Raspberry Pi ran the Mosquitto MQTT broker which subscribed to three topics: ppgFinger, ppgEar, and ecgClient. The ESP32 devices published messages every 50 ms, 10 datapoints per message, to the Raspberry Pi over Wi-Fi. The Mosquitto client would then read the data and write it to an external solid-state drive (SSD) as a CSV file. The overall architecture can be seen in Figure 17. The PPG sensors were then placed on the left ear and finger of the subject, with the ECG pads placed as seen in Figure 17.

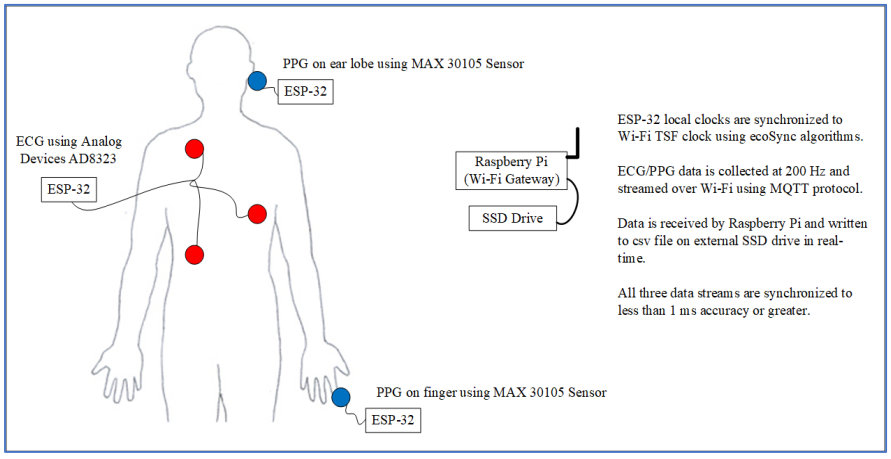

Figure 17. A synchronized WBAN physiological monitoring system.

5.8. Experimental Results

Collected synchronized signals from ECG and two PPGs during slow paced breathing used for relaxation are shown in Figure 18. Pulse travel time for ear and finger locations are clearly identifiable and measurable. Since the signals are precisely synchronized using the ecoSync algorithm, we can use the signals to calculate PAT/PTT and PWV for each heartbeat. The experiment demonstrates that the other sensors running ecoSync, such as smartwatch, smart ring, and Smart Staff, could be synchronized to monitor cardiovascular parameters and assess blood pressure changes and stress throughout the day.

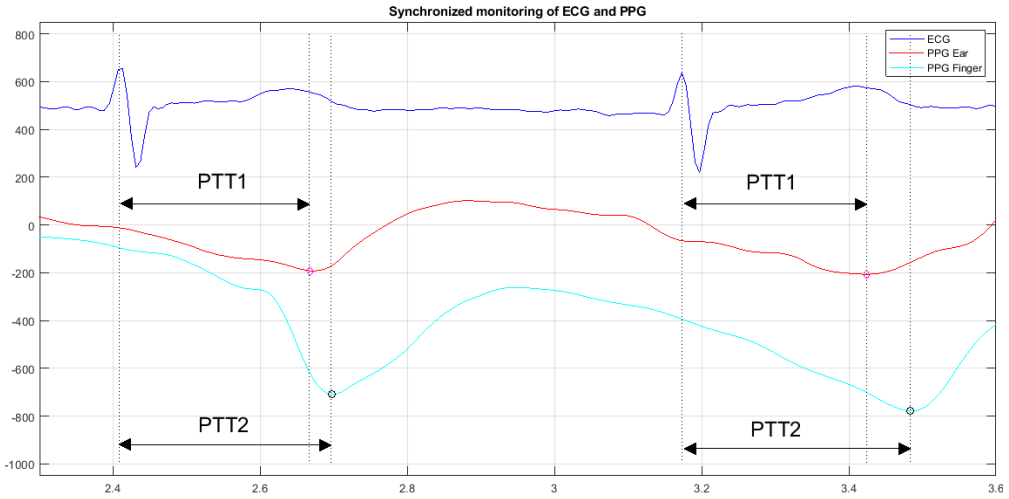

Figure 18. Pulse transit time measured at ear and finger locations.

6. Conclusions

The full potential of IoT applications could be achieved using a synergy of information from multiple sensors in our environment. However, collective processing of data from sensors is possible only if individual sensors are highly synchronized. This is particularly the case in the field of health monitoring applications where individual sensors must be synchronized within one millisecond. Therefore, we developed and implemented an energy-efficient clock optimization and synchronization (ecoSync) algorithm, a time-synchronization algorithm for IoMT devices. The same algorithm would also work on other wireless protocols. We demonstrated that ecoSync provides a sub-millisecond synchronization on standard Wi-Fi networks, and evaluated energy efficiency, synchronization performance, and usability for healthcare campus environments. We also demonstrated the usefulness of the proposed approach for continuous real-time cardiovascular monitoring. The algorithm has been shown to correct time difference errors with high precision and minimal energy usage, making it a valuable tool for healthcare data monitoring.

We believe that the proposed method of Wi-Fi time synchronization for IoMT devices in healthcare campus environments represents an excellent solution for wearable health monitoring applications. While other synchronization protocols and proposals require constant Wi-Fi connectivity with frequent resyncs, our proposed solution can turn off the Wi-Fi radio and stay synchronized to other IoMT devices for large windows of time. We demonstrated that ecoSync can correct the time difference error to ± 42 microseconds with up to an hour between resynchronizations using only 658 millijoules of energy. This is an 87% improvement in time difference error and a 99.93% reduction in energy usage over using TSF alone for synchronization over a one-hour period.

Author Contributions: Methodology, S.P. and E.J.; Software, S.P.; Writing—original draft, S.P.; Writing—review & editing, E.J.; Supervision, E.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to potential intellectual property.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. ESP32 Wi-Fi & Bluetooth MCU I Espressif Systems. Available online: https://www.espressif.com/en/products/socs/esp32 (accessed on 31 July 2023).
2. Jovanov, E.; Milenkovic, A.; Otto, C.; de Groen, P.C. A Wireless Body Area Network of Intelligent Motion Sensors for Computer Assisted Physical Rehabilitation. *J. NeuroEng. Rehabil.* **2005**, *2*, 6. [CrossRef]
3. Coviello, G.; Avitabile, G.; Florio, A. The Importance of Data Synchronization in Multiboard Acquisition Systems. In Proceedings of the 2020 IEEE 20th Mediterranean Electrotechnical Conference (MELECON), Palermo, Italy, 16–18 June 2020; pp. 293–297.
4. Romanov, A.M.; Gringoli, F.; Sikora, A. A Precise Synchronization Method for Future Wireless TSN Networks. *IEEE Trans. Ind. Inform.* **2021**, *17*, 3682–3692. [CrossRef]
5. Haxhibeqiri, J.; Jiao, X.; Aslam, M.; Moerman, I.; Hoebeke, J. Enabling TSN over IEEE 802.11: Low-Overhead Time Synchronization for Wi-Fi Clients. In Proceedings of the 2021 22nd IEEE International Conference on Industrial Technology (ICIT), Virtual Conference, 10–12 March 2021; Volume 1, pp. 1068–1073.
6. Chen, P.; Yang, Z. Understanding Precision Time Protocol in Today's Wi-Fi Networks: A Measurement Study. In Proceedings of the 2021 USENIX Annual Technical Conference, Virtual, 14–16 July 2021.
7. Zinkevich, A.V. ESP8266 Microcontroller Application in Wireless Synchronization Tasks. In Proceedings of the 2021 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, Russia, 17–21 May 2021; pp. 670–674.
8. Avitabile, G.; Florio, A.; Man, K.L.; Zhao, C. A Long-Term Synchronized System for Healthcare. In Proceedings of the 2022 19th International SoC Design Conference (ISOCC), Gangneung-si, Republic of Korea, 19–22 October 2022; pp. 191–192.
9. Avitabile, G.; Man, K.L.; Florio, A. Power Consumption Analysis of a Fractional Approach to BANs Time Synchronization. In Proceedings of the 2021 18th International SoC Design Conference (ISOCC), Jeju Island, Republic of Korea, 6–9 October 2021; pp. 189–190.
10. Zieff, G.; Stone, K.; Paterson, C.; Fryer, S.; Diana, J.; Blackwell, J.; Meyer, M.L.; Stoner, L. Pulse-Wave Velocity Assessments Derived from a Simple Photoplethysmography Device: Agreement with a Referent Device. *Front. Cardiovasc. Med.* **2023**, *10*, 1108219. [CrossRef] [PubMed]
11. Beutel, F.; Van Hoof, C.; Rottenberg, X.; Reesink, K.; Hermeling, E. Pulse Arrival Time Segmentation Into Cardiac and Vascular Intervals—Implications for Pulse Wave Velocity and Blood Pressure Estimation. *IEEE Trans. Biomed. Eng.* **2021**, *68*, 2810–2820. [CrossRef] [PubMed]
12. Jovanov, E. Wearables Meet IoT: Synergistic Personal Area Networks (SPANs). *Sensors* **2019**, *19*, 4295. [CrossRef] [PubMed]
13. Allahi, I.; Khan, B.; Nagra, A.S.; Idrees, R.; Masud, S. Performance Evaluation of IEEE 1588 Protocol Using Raspberry Pi over WLAN. In Proceedings of the 2018 IEEE International Conference on Communication Systems (ICCS), Chengdu, China, 19–21 December 2018; pp. 315–320.
14. Mahmood, A.; Exel, R.; Sauter, T. Performance of IEEE 802.11's Timing Advertisement Against SyncTSF for Wireless Clock Synchronization. *IEEE Trans. Ind. Inform.* **2017**, *13*, 370–379. [CrossRef]
15. Exel, R. Clock Synchronization in IEEE 802.11 Wireless LANs Using Physical Layer Timestamps. In Proceedings of the 2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings, San Francisco, CA, USA, 24–28 September 2012; pp. 1–6.
16. Joulescope JS220: Precision Energy Analyzer. Available online: https://www.joulescope.com/products/js220-joulescope-precision-energy-analyzer (accessed on 11 July 2023).
17. Martin, J.; Burbank, J.; Kasch, W.; Mills, P.D.L. *Network Time Protocol Version 4: Protocol and Algorithms Specification*; Internet Engineering Task Force: Wilmington, DE, USA, 2010.
18. Mills, P.D.L. *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI*; Internet Engineering Task Force: Wilmington, DE, USA, 2006.
19. *IEEE 1588v2*; PTP—Precision Time Protocol. IEEE: New York, NY, USA, 2021.
20. *IEEE Std 80211-2012 Revis. IEEE Std 80211-2007—Redline*; IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications—Redline. IEEE: New York, NY, USA, 2012; pp. 1–5229.
21. Mahmood, A.; Exel, R.; Trsek, H.; Sauter, T. Clock Synchronization Over IEEE 802.11—A Survey of Methodologies and Protocols. *IEEE Trans. Ind. Inform.* **2017**, *13*, 907–922. [CrossRef]
22. Mahmood, A.; Exel, R.; Sauter, T. Delay and Jitter Characterization for Software-Based Clock Synchronization Over WLAN Using PTP. *IEEE Trans. Ind. Inform.* **2014**, *10*, 1198–1206. [CrossRef]
23. Allan, D.W. Time and Frequency (Time-Domain) Characterization, Estimation, and Prediction of Precision Clocks and Oscillators. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control* **1987**, *34*, 647–654. [CrossRef]
24. Yiğitler, H.; Badihi, B.; Jäntti, R. Overview of Time Synchronization for IoT Deployments: Clock Discipline Algorithms and Protocols. *Sensors* **2020**, *20*, 5928. [CrossRef]

25. Karl, H.; Willig, A. *Protocols and Architectures for Wireless Sensor Networks*; Wiley: Hoboken, NJ, USA, 2005; ISBN 978-0-470-09510-2.
26. Aslam, M.; Liu, W.; Jiao, X.; Haxhibeqiri, J.; Miranda, G.; Hoebeke, J.; Marquez-Barja, J.; Moerman, I. Hardware Efficient Clock Synchronization Across Wi-Fi and Ethernet-Based Network Using PTP. *IEEE Trans. Ind. Inform.* **2022**, *18*, 3808–3819. [CrossRef]
27. Coviello, G.; Avitabile, G.; Florio, A.; Talarico, C. A Study on IMU Sampling Rate Mismatch for a Wireless Synchronized Platform. In Proceedings of the 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS), Springfield, MA, USA, 9–12 August 2020; pp. 229–232.
28. Seijo, Ó.; Val, I.; Luvisotto, M.; Pang, Z. Clock Synchronization for Wireless Time-Sensitive Networking: A March from Microsecond to Nanosecond. *IEEE Ind. Electron. Mag.* **2022**, *16*, 35–43. [CrossRef]
29. Coviello, G.; Avitabile, G.; Florio, A.; Talarico, C.; Wang-Roveda, J.M. A Novel Low-Power Time Synchronization Algorithm Based on a Fractional Approach for Wireless Body Area Networks. *IEEE Access* **2021**, *9*, 134916–134928. [CrossRef]
30. Coviello, G.; Avitabile, G.; Talarico, C.; Wang-Roveda, J.M.; Florio, A. Master-Slave Mutual Time Synchronization in a Wireless Body Area Network. In Proceedings of the 2022 IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS), Fukuoka, Japan, 7–10 August 2022; pp. 1–4.
31. Cappelle, J.; Goossens, S.; Strycker, L.D.; der Perre, L.V. Low-Power Synchronization for Multi-IMU WSNs. *IEEE Embed. Syst. Lett.* **2023**, 1. [CrossRef]
32. AD8232 Datasheet and Product Info | Analog Devices. Available online: https://www.analog.com/en/products/ad8232.html#product-overview (accessed on 2 September 2023).
33. SparkFun Single Lead Heart Rate Monitor—AD8232—SEN-12650—SparkFun Electronics. Available online: https://www.sparkfun.com/products/12650 (accessed on 2 September 2023).
34. MAX30102 Datasheet and Product Info | Analog Devices. Available online: https://www.analog.com/en/products/max30102.html (accessed on 2 September 2023).
35. Seeed Studio XIAO ESP32C3—RISC-V Tiny MCU Board with Wi-Fi and Bluetooth5.0, Battery Charge Supported, Power Efficiency and Rich Interface. Available online: https://www.seeedstudio.com/Seeed-XIAO-ESP32C3-p-5431.html (accessed on 2 September 2023).
36. PubSubClient—Arduino Reference. Available online: https://www.arduino.cc/reference/en/libraries/pubsubclient/ (accessed on 2 September 2023).