Received 12 July 2024, accepted 6 August 2024, date of publication 21 August 2024, date of current version 16 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3446668

RESEARCH ARTICLE

MicroSync: Sub-Micro Second Accuracy Wireless Time Synchronization Service

RYOTARO OHARA[1], SHINTARO IZUMI[2,3], (Member, IEEE), SHOYA IMANAKA[1], (Graduate Student Member, IEEE), TETSUO YAMAMURA[2], ISHII TORU[2], AND HIROSHI KAWAGUCHI[2], (Member, IEEE)

[1]Graduate School of System Informatics, Kobe University, Kobe 657-8501, Japan
[2]Graduate School of Science, Technology and Innovation, Kobe University, Kobe 657-8501, Japan
[3]Osaka Heat Cool Inc., 6-20-20 Onoharanishi, Minoh, Osaka 562-0032, Japan

Corresponding author: Shintaro Izumi (shin@cs28.cs.kobe-u.ac.jp)

This work was supported by JST (Japan Science and Technology Agency) under Grant Number JPMJPF2115.

ABSTRACT This paper proposes MicroSync, a high-accuracy and low-power synchronization method based on two ideas: a hybrid timer and communication scheduling. The hybrid timer is implemented by combining two types of timers: a real-time clock and high-frequency clock timers. While high-frequency clock timers offer high resolution at the cost of increased power consumption, real-time clocks operate at lower frequencies, providing power efficiency but lower time resolution. The combination of these two timers results in a synchronized timer with low power consumption and high accuracy. Furthermore, we proposed two types of communication scheduling using this hybrid timer: high-accuracy and low-power scheduling. High-accuracy scheduling leverages Slave Latency, a feature of Bluetooth Low Energy (BLE), to minimize communication frequency, reduce the time gap between communication and synchronization, and prevent error accumulation caused by frequency drift. Both synchronization schemes were implemented on a commercially available Bluetooth transceiver IC. The evaluation results show that the high-accuracy scheduling achieves 400-ns synchronization accuracy with less than 300 μW power consumption. The low-power scheduling also achieves 1-μs synchronization accuracy with less than 150 μW power consumption under same conditions.

INDEX TERMS Bluetooth, synchronization, protocols, receivers, Internet of Things, Bluetooth low energy.

I. INTRODUCTION

Internet of Things (IoT) technologies, such as smart homes and sensor networks, have rapidly gained popularity in recent years. In such systems, several devices connected to a network work together to collect data and control the system. Time synchronization between devices is essential for time-series processing of data sensed by multiple devices and for controlling transducers and actuators.

A challenge in time synchronization systems is the trade-off between power savings and accuracy. Low power consumption is an important aspect for many IoT systems, as IoT devices are distributed in various environments and many operate on batteries or energy harvesting systems. Specifically, in environments where external power supply or frequent battery replacement is not feasible, such as for rotating components like wind turbines or crankshafts [1], power-saving becomes essential. Accurate time synchronization is a crucial factor for sensors and actuators that require real-time control. For example, in systems that handle ultrasonic waves, such as ultrasonic position measurement [2] and ultrasonic sensing using microphone arrays [3], synchronization on a sub-microsecond basis is necessary to rearrange the reception timing between microphones and measure the elapsed time between transducers and microphones. Additionally, when measuring vital data, such as human motion sensing and electromyography (EMG) using distributed sensors, wireless time synchronization technology is required to synchronize measurement timing. Given that distributed sensors are often powered by batteries, low-power time

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

synchronization technology enables long-term measurements. Hence, low-power and high-accuracy time synchronization is one of the most crucial services in IoT systems.

The motivation for this study stems from the critical need to address the trade-off between power efficiency and synchronization accuracy in IoT systems. To date, various studies on time synchronization in IoT systems using wireless communication technologies. In those systems, there is a trade-off between time synchronization accuracy and power consumption, because high-frequency timers with high-accuracy consume more power. This research proposes a new time synchronization method, MicroSync, based on BLE, a Bluetooth low-power standard. The core hypotheses of the proposed MicroSync are twofold. First, by combining a high-accuracy, high-power consumption timer with a low-power, low-resolution timer, it is possible to maintain high time synchronization accuracy without compromising energy efficiency. Second, further power savings can be achieved by optimally scheduling time synchronization communications to align with BLE communication timings. Based on these hypotheses, we have successfully developed a time synchronization system, MicroSync, that effectively balances low power consumption with high-accuracy.

Bluetooth Low Energy (BLE), a low-power wireless communication protocol standardized by the Bluetooth Special Interest Group, has emerged as a key IoT technology since its release in the Bluetooth 4.0 [4], [5]. In typical Bluetooth connections, a single computer or smartphone acts as the Central, and multiple Peripherals such as mice and sensor devices are connected. In the past decade, several synchronization technologies based on the BLE have been studied [1], [6], [7], [8], [9], [10], [12], and some research has been conducted using these technologies [1], [13]. However, few studies have focused on reducing power consumption in BLE-based time synchronization systems. In addition, to perform time synchronization, communication needs to synchronize with each other; however, a large power consumption instantaneously to drive the power amplifier (PA) and the low-noise amplifier (LNA).

Fig. 1 shows an overview of MicroSync, which is realized by two power reduction technologies: hybridization of timers and scheduling of synchronization. The hybridization technique combines a high-frequency clock (HFCLK) timer and real time clock (RTC). The proposed scheduling uses Slave Latency, a power-saving function of BLE, and pipelining, which can reduce the number of communications required for time synchronization. The three main points summarize the contributions of our study.

Two types of scheduling are proposed: high-accuracy and low-power scheduling. The high-accuracy scheduling achieved a mean absolute error of 272 ns and a standard deviation of 319.2 ns at 252.1 μW. Low-power scheduling achieved a mean absolute error of 796.4 ns and standard deviation of 990.4 ns at a power consumption of 141.8 μW.

Furthermore, we proposed a hybrid approach to timer synchronization, achieving power savings with negligible degradation compared to an HFCLK timer implementation. In high-accuracy scheduling, the power reduction was 90.9% compared to that of the HFCLK timer. Low-power scheduling also achieved a 94.8% power reduction compared with the HFCLK timer implementation.

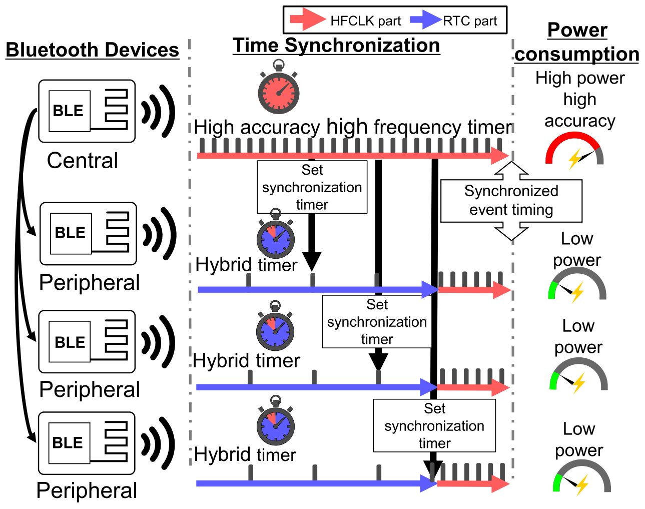


FIGURE 1. Overview diagram of the MicroSync synchronization algorithm.

In the hybridization of timers, the running time of the HFCLK timer using crystal as the clock source was sufficiently short to suppress the increase in power consumption when the frequency of the HFCLK timer was increased.

The synchronization method we developed was implemented ion nRF52840, which is manufactured by Nordic Semiconductor. Furthermore, the program was built on Soft Device, a firmware released by Nordic Semiconductor, and does not directly program radio binaries; therefore, it can be used in many countries without worrying about compliance with technical standards.

The remainder of this paper is organized as follows. Section II introduces the previous work and clarifies the contributions of this study. In Section III, we explain the hybridization of timers, which is a power-saving technology used in this study, and the Slave Latency of BLE. Section IV describes the two proposed MicroSync synchronization schemes: a high-accuracy scheme and a low-power scheme, and their hybrid timer implementation. Section V describes the experimental setup and conditions used to evaluate the performance of the proposed method. Section VI describes the experimental results and analyses. Section VII describes the superiority of the proposed method over conventional methods based on experimental results. Section VIII discusses future work based on the experimental results. Finally, section IX presents the conclusions.

II. PREVIOUS WORKS

This section describes the various time synchronization methods proposed in the field of IoT.

The global positioning system (GPS) allows receivers to determine their current locations by receiving radio signals from multiple GPS satellites. Because GPS obtains time from atomic clocks installed on satellites, it allows the determination of absolute time with an accuracy of a few microseconds. This is applicable even for devices that cannot be equipped with expensive atomic clocks provided that the internal system clock can be accessed [14]. However, accessing GPS satellites requires that the device always receive signals from multiple GPS satellites, which is not optimal for IoT systems in indoor environments, such as smart homes, or for sensor networks installed in various locations under different conditions. In addition, the GPS also consumes a significant amount of power, which makes it unsuitable for systems designed to run on batteries for several years.

The Network Time Protocol (NTP) is a widely used synchronization protocol in IoT devices running on PCs and IP networks, and has excellent synchronization accuracy and robustness. Because it runs on IP networks, the implementation cost of the protocol stack is high owing to routing and other issues, and the power consumption tends to increase. Therefore, they are unsuitable for sensor networks and IoT systems that are not externally powered.

The Precision Time Protocol (PTP) provides more accurate time synchronization than NTP over IP networks. PTP is available over wired communication standards such as Ethernet (IEEE802) and compatible protocols such as Wi-Fi (IEEE802.11) [15]. A PTP can achieve sub-microsecond accuracy under ideal conditions and requires hardware PTP support in an Ethernet physical layer (PHY). PTP gained attention as the preferred network changed from CAN to Ethernet, and microcontroller implementations became available. In a previous study [16], a microcontroller with an Ethernet PHY and hardware support for PTP achieved synchronization with sub microsecond accuracy. As a wireless example, a prior study [17] implemented the PTP protocol on a microcontroller ESP32 with Wi-Fi. This implementation achieved a time synchronization accuracy of 15 μs without hardware PTP support, and PCs or single-board computers achieved sub-microsecond accuracy under Wi-Fi [18]. In a previous study [19], an FPGA implementation of PTP was proposed using low-speed, low-data-rate Bluetooth, and a synchronization accuracy of 15 ns was realized.

Reference Broadcast Synchronization (RBS) [20] is an algorithm that synchronizes the time between neighboring nodes instead of synchronizing them to a Universal Standard Time (UTC), such as GPS. The reference node broadcasts a synchronization packet to all the nodes, and the receiving nodes exchange their local time when they receive the synchronization packet and calculate a correction value based on the error in their received time to achieve accurate time synchronization. However, there is a tradeoff: as the number of nodes increases, the accuracy improves, but the amount of communication also increases.

The Timing-sync Protocol for Sensor Network (TPSN) [21] algorithm achieves time synchronization through bidirectional communication such as NTP. In this algorithm, the receiving node is synchronized with the sending node. Synchronization is performed by first constructing an entire tree with the reference node as the root. The parent and child nodes of the tree then communicate with each other to perform time synchronization using multiple timestamps, which are generated at the MAC layer to reduce communication delays.

Some Bluetooth-based implementations require additional hardware for Bluetooth devices, whereas others do not. In addition, BLE has a generic access profile (GAP) for broadcast communication, such as advertising, and generic attributes (GATT) for one-to-one communication. Two types of communication methods are used for synchronization: one that establishes communication between the Central and Peripherals and the other that achieves time synchronization through broadcast advertising.

The current analysis is a typical method that uses additional hardware for time synchronization [6], [8], [9], [11]. In BLE communication, power pattern analysis, in which a shunt resistor is inserted into the power supply line of the BLE device and the potential difference is measured by an amplifier, is another method for determining the timing of transmission, in addition to the method using firmware interrupts. The current waveform of a BLE device includes the features of the transmitting and receiving amplifiers, from which the transmission current of the PA of the BLE device, the receiving current of the LNA, and other communication details can be accurately determined [6], [8]. In [11], synchronization with a variance of 0.9 μs was achieved by power consumption analysis using a shunt resistor and comparator.

In BLE, there are two types of communication: GAP, such as advertising communication, where data are broadcasted; and GATT, where bidirectional communication is performed by establishing a connection. CheepSync [10] and BlueSync [7] are methods for achieving time synchronization through advertising. Advertising requires only one-way communication. The time-source device only needs to consider the transmission power and does not need to consider the increase in power as the number of receivers increases. CheepSync combines synchronization algorithm that uses advertising and compensates for clock drift to achieve an average accuracy of 10 μs. BlueSync is another method for achieving time synchronization using advertising communication.

Reference [12] presented a method for time synchronization using GATT. In particular, the Central is connected to the multiple Peripherals via GATT communication, and the Central side uses the reference time sent from the Peripheral to calculate the synchronization time, which is communicated to the Peripheral, thereby realizing a synchronization time of 20 μs with an RTC of 32.768 kHz.

To date, there are few studies on time synchronization technology that focus on power savings. ecoSync, a low-power synchronization scheme, is implemented on ESP32 microcontroller that supports Wi-Fi. ecoSync realizes synchronization with the CPU clock by resetting the timing synchronization function counter internal to the Wi-Fi module. The results of an actual experiment using ESP32 DEVKIT V1 demonstrate a synchronization accuracy of 42 μs and power consumption of 182 μW at a resynchronization interval of 60 min.

In a previous study [22] a power-saving time synchronization method, which utilizes Bluetooth, was reported. This study focuses on Wireless Body Area Networks (WBANs) and assumes data collection from body-worn accelerometers. The employed network topology is a single-hop star topology, capable of supporting up to eight Peripheral devices. The synchronization algorithm uses a simplified method based on FSTP. When the resynchronization interval is set to 1 s, the synchronization accuracy is 4.42 ms and power consumption corresponds to 3.34 mW. When the synchronization interval is set to 60 s, the synchronization accuracy is 220 ms and power consumption is 55 μW. This synchronization accuracy is sufficiently higher than the inertial measurement unit's (IMU) sampling rate of 100 Hz, indicating that it is sufficiently fast for target application.

Reducing the number of transmissions and power consumption via scheduling is a well-studied topic in the field of sensor networks. Microcontrollers with BLE modules require mA-scale currents to drive LNA and PA. Reducing the number of communications contributes significantly to power savings. Compared with single-hop method, sensor networks have a larger number of nodes and more complex network topology. Therefore, communication collisions are likely to occur and power is likely to increase [23]. Certain studies involving FDAS [24] and LPSRS [25] use scheduling to avoid communication collisions and reduce the number of communications. This in turn reduces power consumption. SLES [26] reduces the number of communications by 22% when compared with conventional synchronization algorithms such as EERS [27] and FADS. These algorithms focus on reducing overall system power by minimizing the number of communications, with limited discussion on the power consumption of individual devices.

III. POWER SAVING METHOD

The proposed MicroSync uses two techniques for power reduction: the hybrid control of timers and the utilization of Slave Latency. The two techniques are described in this section.

A. HYBRID TIMER

Most microcontrollers with BLE have two types of timers: HFCLK timers for high-accuracy time measurement and a real-time clock (RTC) for second to day long-time measurement. HFCLK timers are suitable for use in millisecond to microsecond real-time events and have high resolution in the time direction; however, they consume more power owing to their high clock frequencies. Furthermore, if an external crystal is used, the power consumption increases.

RTC is designed to measure long periods of time with very low power consumption. They are suitable for measuring long periods of time at millisecond, minute, and day levels. In addition, RTCs are often intended for use when the processor is asleep; therefore, they have a low clock frequency and coarse temporal resolution but consume very low power as it draws a very small current, from a few microamperes to some tens of microamperes.

By combining these two types of timers with different time resolutions, power savings and highly accurate time synchronization can be achieved. Furthermore, the power consumption of a timer includes not only that of the timer counter itself but also that of the crystal or RC oscillator that serves as the clock source. Therefore, the power consumption can be reduced by turning off the power supply to the crystal oscillator, phase-locked loop (PLL), and other Peripheral devices.

Here, we describe a hybridization method between HFCLK and RTC timers. Generally, switching the system clock source from an RC oscillator to an external crystal results in a current flow of several hundred microamperes, even if the timer counter is stopped. Therefore, to save power, it is effective not only to stop counting the clock, but also to disable the crystal oscillator, switch the internal system clock to the RC oscillator, and restart the crystal oscillator and PLL only when necessary. However, the crystal oscillator and PLL are unstable immediately after start-up, requiring a warm-up time until the clock stabilizes.

Fig. 2 shows the switching sequence from RTC to high-frequency timer, where the hybridization of RTC and high-frequency timer is realized by three control events. First, the high-frequency clock (HFCLK) warmup event starts up the crystal oscillator and Peripheral circuits. Next, after a warm-up time, the HFCLK startup event is triggered by an RTC interrupt and starts the high-frequency timer counting. During HFCLK running, the HFCLK is either interrupted by a high-frequency timer or by a time measurement. In the case of high-frequency timer interrupt, high-accuracy interrupt is possible, and in the case of time measurement, time can be measured with the resolution of the HFCLK. In this case, the RTC value at the time of the startup event is recorded and used as the starting point for the HFCLK time calculation. Finally, the HFCLK is released and shutting-down to reduce power consumption. This event is generated in the high-frequency timer Event or by the RTC. This sequence of operations is how the hybrid's timer processing is accomplished.

The mechanism of timer hybridization is described in detail. Please note the following three points.

- The difference between an RTC timer and HFCLK timer is the resolution as opposed to precision of a single clock.
- The timer value is calculated with the resolution of the HFCLK timer.

The timer value is calculated as a real number and rounded off before it is stored in the timer register.

The left side of Fig. 3 shows the time recorded using the hybrid timer. First, an RTC timer interrupt is triggered. The value of the RTC timer is recorded as RTC_{base}. Immediately thereafter, HFCLK timer is removed and started. If an external interrupt occurs while HFCLK timer is operating, then the value of HFCLK timer is recorded as $HFCLK_{base}$. The absolute time at which an external interruption occurs is given

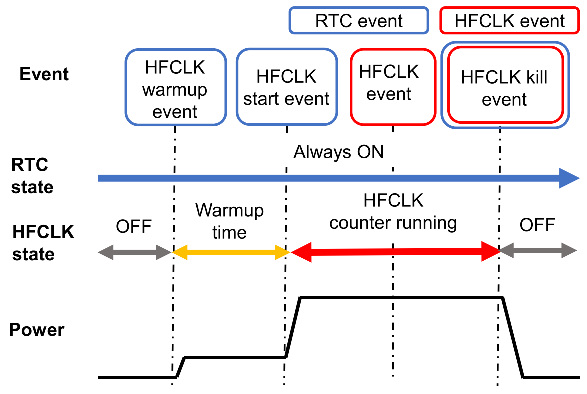


FIGURE 2. Hybrid control of HFCLK timer and RTC.

by the following formula.

$$T_{\mathrm{HFCLK}} = T - round\,(T_{\mathrm{RTC}}\,(F_{\mathrm{HFCLK}}/F_{\mathrm{RTC}})) \quad (1)$$

A disadvantage of this method is that the HFCLK timer must be started before any external interruption occurs.

The right side of Fig. 3 shows a timer interrupt using a hybrid timer. In this system, the timer value is expressed as an absolute value. Therefore, the interrupt timing is related to the base point T_{base}. Here, T indicates the time from T_{base} to the interrupt timing. In the hybrid timer, the interrupt time T must be decomposed into T_{RTC} and T_{HFCLK} and set in the timer register. The following equation represents the RTC time part counted by the RTC.

$$T_{\mathrm{RTC}} = \mathrm{round}\,(T/\,(F_{\mathrm{HFCLK}}/F_{\mathrm{RTC}})\,-\mathrm{margin}) \quad (2)$$

A margin is introduced to ensure that T_{HFCLK} does not become zero. This can occur when $T/\,(F_{\mathrm{HFCLK}}/F_{\mathrm{RTC}})$ is perfectly divisible, leaving no remainder.

The time recorded by HFCLK timer is calculated using the following equation:

$$T_{\mathrm{HFCLK}} = T - round\,(T_{\mathrm{RTC}}\,(F_{\mathrm{HFCLK}}/F_{\mathrm{RTC}})) \quad (3)$$

The part counted by HFCLK timer is converted from remaining time T_{RTC} counted by RTC timer to the resolution of HFCLK timer and subtracted from T. Given that the value actually set as RTC interrupt timing is relative to T_{base}, it is $\mathrm{RTC_{base}} + T_{\mathrm{HFCLK}}$, and the value set in HFCLK timer is T_{HFCLK}. Fig. 2 shows the switching sequence from RTC to HFCLK timer, where the hybridization of the RTC and HFCLK timers is realized by three control events. The HFCLK warm-up event begins with a crystal oscillator and Peripheral devices. Next, after the warm-up time, the HFCLK start-up event is triggered by an RTC interrupt and HFCLK is started.

B. SLAVE LATENCY

Slave Latency is a power-saving function in the BLE standard. In BLE, GATT communication is performed after the communication between the Central and Peripheral devices is established. This communication interval is known as

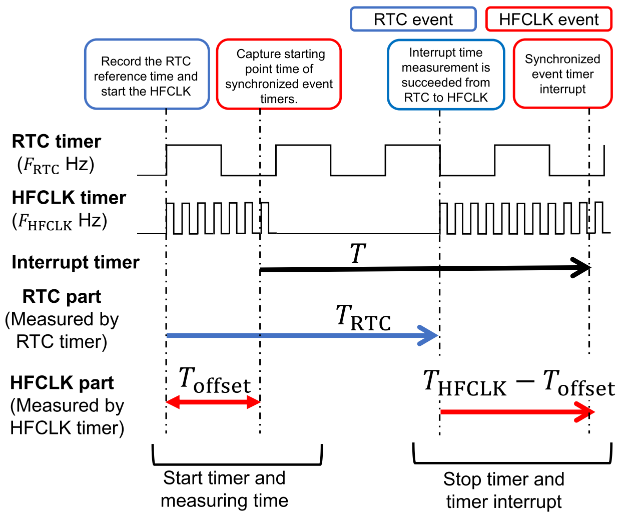

RTC event
HFCLK event
Record the RTC reference time and start the HFCLK
Capture starting point time of synchronized event timers.
Interrupt time measurement is succeeded from RTC to HFCLK
Synchronized event timer interrupt
RTC timer (F_{RTC} Hz)
HFCLK timer (F_{HFCLK} Hz)
Interrupt timer
T
RTC part (Measured by RTC timer)
T_{RTC}
HFCLK part (Measured by HFCLK timer)
T_{offset}
$T_{\mathrm{HFCLK}} - T_{\mathrm{offset}}$
Start timer and measuring time
Stop timer and timer interrupt

FIGURE 3. Operation of the decomposed timers T_{RTC} and T_{HFCLK}. Operation of the decomposed timers T_{RTC} and T_{HFCLK}.

the Connection Interval, which can be set from 7.5 ms to 4 seconds.

Shorter Connection Intervals result in shorter delays between when the application layer communication data are ready and when they are transmitted. In other words, communication delays are reduced. However, the number of communications per unit of time increases, resulting in increased consumption. Conversely, if the Connection Interval is long, the number of communications is reduced and power is saved; however, the throughput worsens.

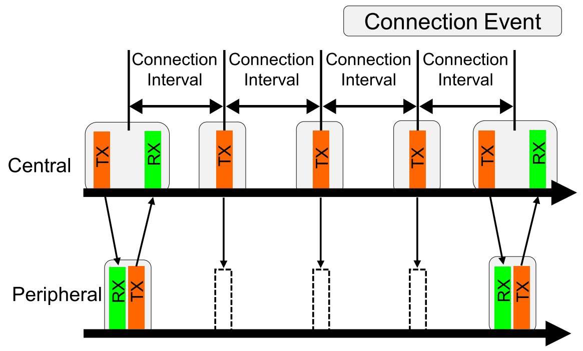


FIGURE 4. Reduction in communication frequency by three in the case of slave latency.

Slave Latency is a mechanism that avoids these problems by reducing power consumption when there are no data to send at the Peripheral side. Fig. 4 shows an overview of the Slave Latency. The squares at the top of the timeline indicate communication events. The green and orange rectangles represent receiving and transmission, respectively. Communication occurs at intervals of the Connection Interval, but Slave Latency makes it possible to ignore communication from the center a specified number of times. For example, if the Slave Latency is set to four, communication can be omitted when no data are ready for transmission, thereby reducing power consumption by a maximum of 1/4 in Peripheral.

IV. COMMUNICATION SCHEDULING

Figure 5 presents an overview of the scheduling algorithms proposed in this paper. (a) represents the high-precision scheduling, where synchronization timing is calculated and shared through three communications. (b) represents the low-power scheduling, which pipelines the method used in (a). These algorithms will be described in detail in subsequent sections.

This section describes the MicroSync synchronization algorithm. Subsection A describes the communication and data exchange that form the basis of the theoretical synchronization algorithm. Subsection B describes the communication scheduling implemented for highly accurate synchronization using the synchronization algorithm detailed in Subsection A. Subsection C describes the pipelining implemented for this synchronization algorithm to achieve power saving. Subsection D describes the simple correction algorithm.

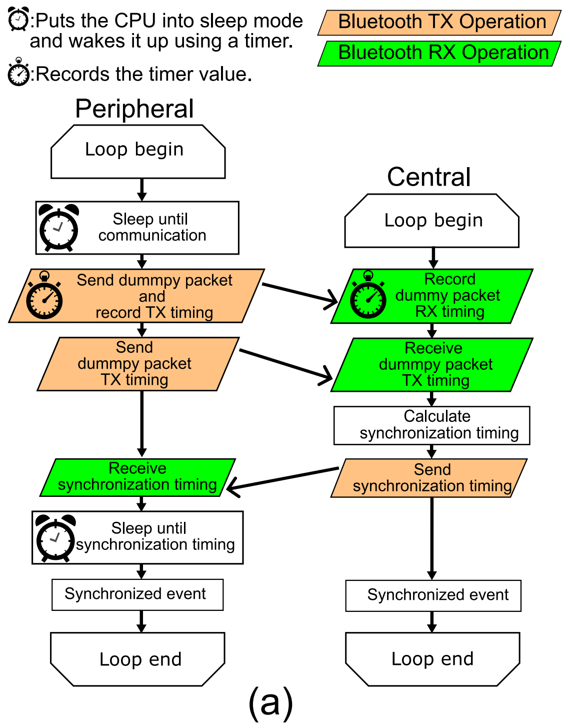
A. SYNCHRONIZATION SEQUENCE

The synchronization sequence of the proposed algorithm is illustrated in Fig. 6. Synchronization consists of three communications: sending a dummy packet from the Peripheral to the Central device, sending a timer value for sending the dummy packet, and returning a timer value for timing synchronization. These communications were realized in three Peripheral events, A_n, C_n, and F_n, and three Central events, B_n, D_n, and E_n, as shown in Fig. 6.

First, a dummy packet is sent from the Peripheral event A_n. The Peripheral recorded the timer value for this event to determine the dummy packet transmission timing. After a slight delay, this dummy packet is received at the Central device in event B_n. This dummy packet reception time was recorded from the timer value of the Central device. At event C_n, the Peripheral device sends the recorded dummy packet transmission timing for event A_n. Because the timing of the send/receive events must be measured at the lower layer, the transmission timing of the first packet cannot be obtained at the application layer. Thus, two packets were used in this sequence.

The Central device can calculate the correct synchronization timing using this received packet at event D_n. At this time, it can know the timer value that the Peripheral has and recognize the deviation and the correction. The estimated synchronization timing is sent back to the Peripheral device from Central device in event E_n. In event F_n, the Peripheral sets the timer interrupt based on the packet received from the Central device. Then, at the time of synchronization, an interrupt is generated in the Peripheral as well as Central devices.

This synchronization algorithm is an improvement over the communication algorithms presented in prior studies [12]. The improvement is that we modified it so that dummy


:Puts the CPU into sleep mode and wakes it up using a timer.
:Records the timer value.
Bluetooth TX Operation
Bluetooth RX Operation
Peripheral
Loop begin
Sleep until communication
Send dummpy packet and record TX timing
Send dummpy packet TX timing
Receive synchronization timing
Sleep until synchronization timing
Synchronized event
Loop end
Central
Loop begin
Record dummy packet RX timing
Receive dummpy packet TX timing
Calculate synchronization timing
Send synchronization timing
Synchronized event
Loop end
(a)

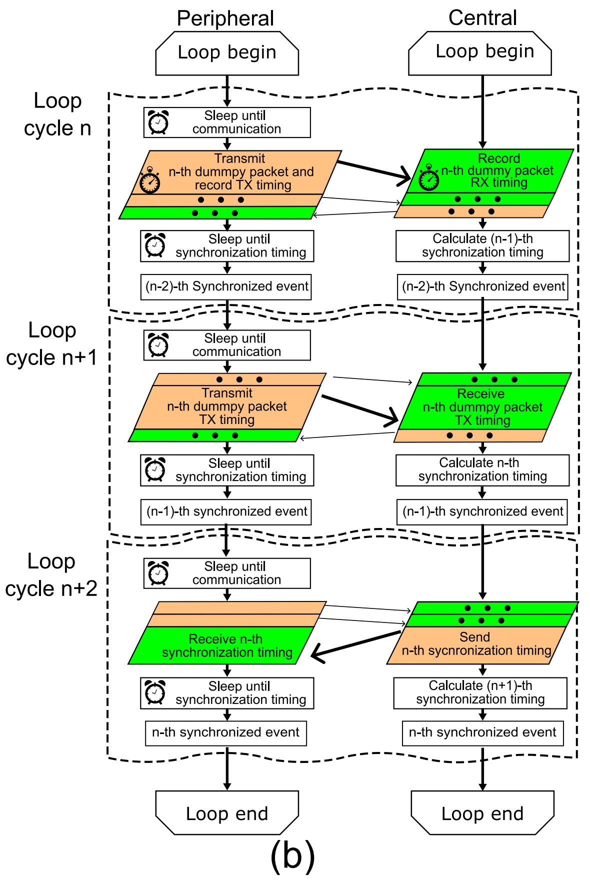

Peripheral
Central
Loop begin
Loop cycle n
Sleep until communication
Transmit n-th dummpy packet and record TX timing
Record n-th dummy packet RX timing
Sleep until sychronization timing
Calculate (n-1)-th sychronization timing
(n-2)-th Synchronized event
Loop cycle n+1
Transmit n-th dummpy packet TX timing
Receive n-th dummpy packet TX timing
Calculate n-th synchronization timing
(n-1)-th synchronized event
Loop cycle n+2
Receive n-th synchronization timing
Send n-th sycnronization timing
Calculate (n+1)-th synchronization timing
n-th synchronized event
Loop end
(b)

FIGURE 5. Flowchart of synchronization scheduling algorithm (a) high-accuracy scheduling, (b) low-power scheduling.

packets are sent at any time from the Central side instead of the Central side requesting the transmission of dummy

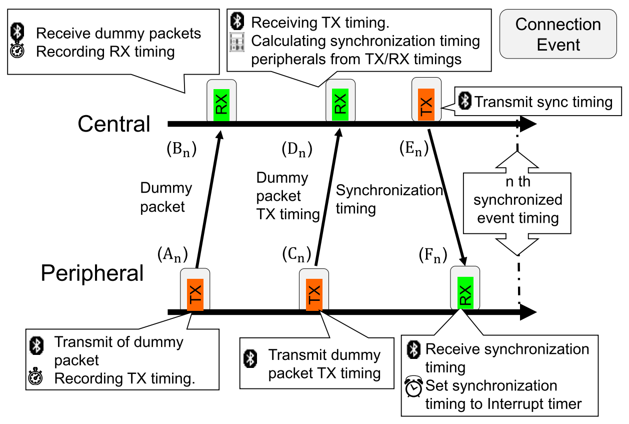


FIGURE 6. MicroSync time synchronization sequence.

packets to the Peripherals. Consequently, the number of communications decreased from four to three. This reduces the power of the LNA associated with reception on the Peripheral side. These sequences can be performed in the same manner as when multiple Peripherals exist. The performance evaluation showed that they could be executed using three Peripherals.

Figure 7 shows sequence of MicroSync implementation. The implementation of this system is as follows: This algorithm is realized by combining many functions such as event handlers and interrupt processing, from (a) to (i). Fig. 7 illustrates the sequence of the system. The sequence begins on the Peripheral side. First, function (a) is invoked by timer interruptions. This function reserves a dummy packet for transmission toward the Central region. Subsequently, in communication (b), the dummy packet reserved in (a) is transmitted by SoftDevice. With the transmission, PA pin rises and the interrupt function (c) is invoked by a GPIO interrupt. Given that the transmission reservation at the application layer differs from the actual transmission timing, it is necessary to measure the timing at the lower layer. The Soft Device has a function to externally output the timing of PA operation during BLE transmission as a GPIO. In the interrupt function (c), the timer value at the time of the GPIO interrupt is recorded as X_{tp}.

On the Central side, the dummy packet is received, LNA pin rises, and the time is recorded in the interrupt function (d). Subsequently, the receiver handler (e) on the Central side is invoked. The sender of the packet is not known until the packet content is examined by the receiver handler. The receiving time recorded in (d) is associated with the sender's Peripheral.

On the Peripheral side, in communication (f), the transmission time X_{tp} reserved for function (c) is transmitted. Additionally, it reserves the synchronized event timing S for transmission to the Peripheral. During communication (h), event timing S is transmitted. On the Peripheral side, the receiver handler (j) is invoked. In this handler, the synchronization timing sent in communication (h) is set to the interrupt timer, and a synchronized event is then generated.

B. HIGH-ACCURACY SCHEDULING

This section describes the implementation of synchronization for high-accuracy scheduling using Slave Latency on BLE, calculation of synchronization timing, and hybridization of timers.

Our scheduling algorithm is driven and realized by several interruption events: communication interrupts, timer interrupts, and GPIO interrupts. A communication interrupt is used to detect Bluetooth communication events. Timer interrupts are triggered by timers such as RTC and HFCLK. Additionally, in this system, the firmware function detects the communication timing output from the GPIO by using GPIO interrupts on other GPIO pins.

First, in (A_n) in Fig. 8 a dummy packet transmission event is generated by a timer interrupt in the application layer. In BLE, communication timing is restricted by the Connection Interval, and data cannot be sent at arbitrary times. Therefore, the transmission is reserved between the Connection Event to be transmitted and the previous Connection Event.

The interrupt timing at this time is set to X_{tp}. This X_{tp} is reserved for transmission to Central. The event in (B_n) records the timing of dummy packet reception on Central side. Furthermore, Central device records the timing of the dummy packet reception using a Soft Device interrupt. When a dummy packet is received, the LNA notification pin is set and a GPIO interrupt is generated. The dummy packet receiving timing is recorded within this GPIO interruption. Immediately after this, a Bluetooth receiver interrupt is generated, which is associated with the reception timing of the Peripheral and dummy packets in this function. The event in (C_n), X_{tp} is sent to the Central and received in event (D_n). In (D_n), timing from the dummy packet reception timing on the Central side to the interruption is added, starting with the transmission timing of the dummy packet sent from the Peripheral. The synchronization timing transmission is then reserved to return to Peripheral.

In this system, the synchronization cycle is 1 s. In high-accuracy scheduling, three communications are performed for each synchronization. Hence, the Slave Latency is assumed to be 20 ms. Therefore, setting Slave Latency to 47 times results in a synchronization period of 1 s per synchronization.

The calculation of the synchronization timing indicated in event D_n in Figs. 6 and 7 is detailed in Fig. 9 First, a dummy packet is sent, and its transmission timing is recorded by the interrupt generated by the detection of the PA operation in the transmitter circuit. The Central device records the time using an interrupt generated by the LNA operation of the receiver circuit. Because these two timings do not coincide perfectly, a delay d occurs, as shown in Fig. 9. Here, t_1 denotes the elapsed time from the previous synchronization event to the current interrupt, t_2 represents the interval from the LNA to the reception of the interrupt, and L denotes the synchronization cycle, respectively. Using these terms, the time to the next synchronization on the Central side can be calculated. The synchronization time S is calculated at the Peripheral by

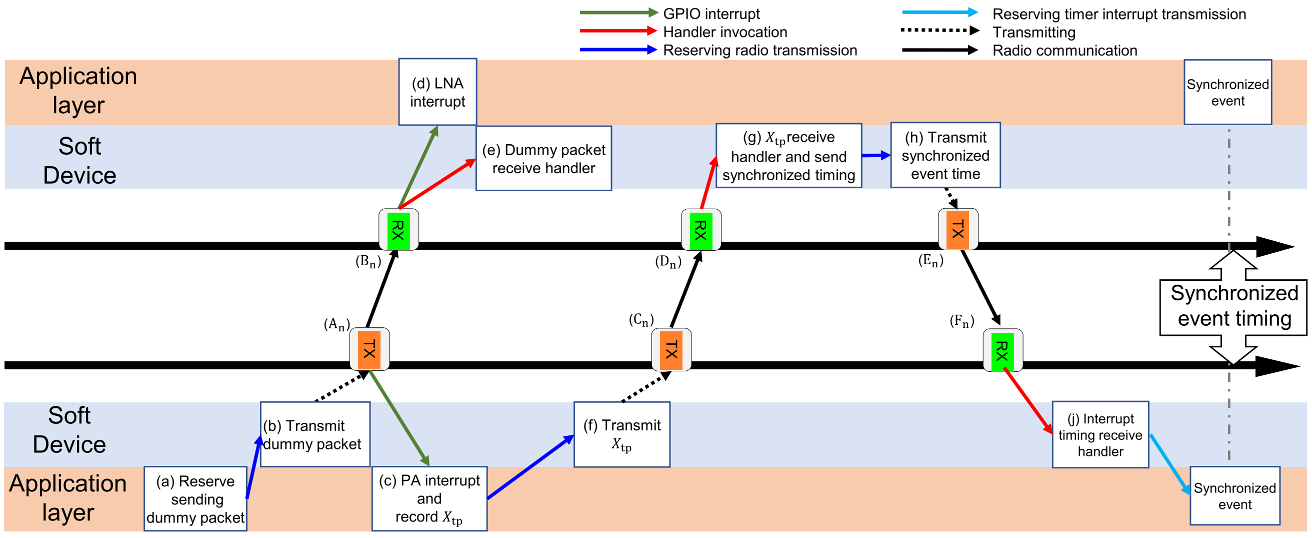

GPIO interrupt
Handler invocation
Reserving radio transmission
Reserving timer interrupt transmission
Transmitting
Radio communication
Application layer
Soft Device
(d) LNA interrupt
(e) Dummy packet receive handler
(g) X_{tp} receive handler and send synchronized timing
(h) Transmit synchronized event time
Synchronized event
Synchronized event timing
(b) Transmit dummy packet
(a) Reserve sending dummy packet
(c) PA interrupt and record X_{tp}
(f) Transmit X_{tp}
(j) Interrupt timing receive handler
RX
TX

FIGURE 7. Software sequence diagram for peripheral-central communication synchronization.

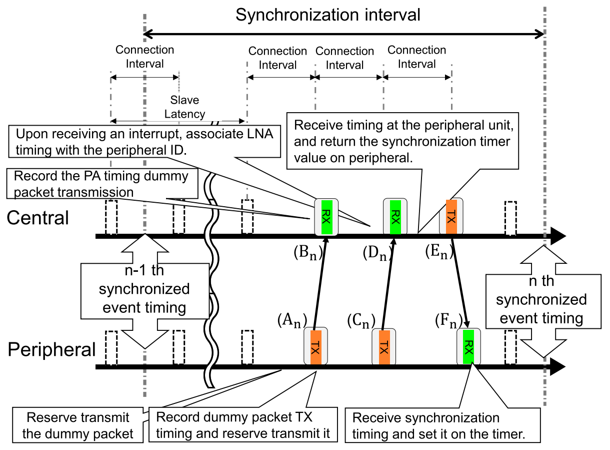

Synchronization interval
Connection Interval
Slave Latency
Upon receiving an interrupt, associate LNA timing with the peripheral ID.
Receive timing at the peripheral unit, and return the synchronization timer value on peripheral.
Record the PA timing dummy packet transmission
Central
Peripheral
n-1 th synchronized event timing
n th synchronized event timing
Reserve transmit the dummy packet
Record dummy packet TX timing and reserve transmit it
Receive synchronization timing and set it on the timer.

FIGURE 8. Implementation of high-accuracy scheduling and power saving by slave latency.

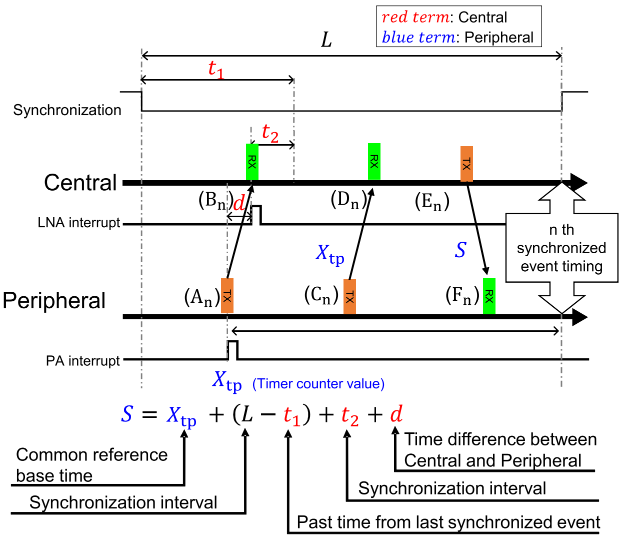

red term: Central
blue term: Peripheral
Synchronization
Central
LNA interrupt
Peripheral
PA interrupt
X_{tp} (Timer counter value)
n th synchronized event timing
$S = X_{tp} + (L - t_1) + t_2 + d$
Common reference base time
Synchronization interval
Past time from last synchronized event
Synchronization interval
Time difference between Central and Peripheral

FIGURE 9. Calculation of synchronization timing in high-accuracy scheduling.

adding the time until synchronization to the transmission time X_{tp}, which is transmitted from the Central to the Peripheral. Although the delay d varies depending on the device circuit characteristics and communication conditions, its variation is sufficiently small for the desired synchronization accuracy. Therefore, d can be measured in advance and hard-coded as a fixed value.

The value of S is given by the following equation:

$$S = X_{tp} + (L - t_1) + t_2 + d \tag{4}$$

Equation (4) defines the timer value S for the next Synchronized Event of the Peripheral. Here, X_{tp} is the common reference timing for the Central and Peripheral regions. L is the synchronization interval and t_1 is the time since the last Synchronized Event. The term $(L - t_1)$ is the time remaining until the next Synchronized Event from the point of the receiver interrupt. Variable t_2 is the time between the GPIO interruption triggered by the LNA and its detection, and d represents the timing error in X_{tp} between the Central and Peripheral devices. In summary, the timer setting for a Synchronized Event at a Peripheral location comprises X_{tp} plus $(L - t_1)$, t_2, and d.

Figure 10 presents an overview of the hybridization of high-accuracy scheduling. In our scenario, the Central device assumed that the power consumption constraints were negligible or sufficiently modest and that HFCLK was always running. Therefore, the hybrid operation described here was mainly adapted to Peripherals. Mission timing, and the counter value of the RTC was recorded at this time. When an interrupt caused by the PA operation occurred, the HFCLK timer counter value at that time was recorded and the HFCLK

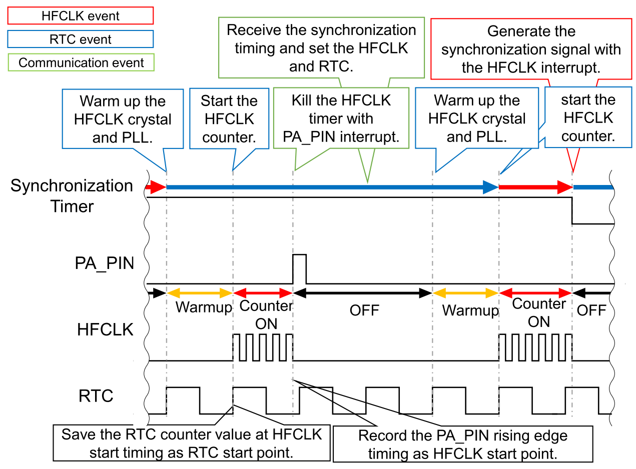

HFCLK event
RTC event
Communication event
Receive the synchronization timing and set the HFCLK and RTC.
Generate the synchronization signal with the HFCLK interrupt.
Warm up the HFCLK crystal and PLL.
Start the HFCLK counter.
Kill the HFCLK timer with PA_PIN interrupt.
Warm up the HFCLK crystal and PLL.
start the HFCLK counter.
Synchronization Timer
PA_PIN
HFCLK
RTC
Warmup
Counter ON
OFF
Warmup
Counter ON
OFF
Save the RTC counter value at HFCLK start timing as RTC start point.
Record the PA_PIN rising edge timing as HFCLK start point.

FIGURE 10. Overview of hybridization of high-accuracy scheduling.

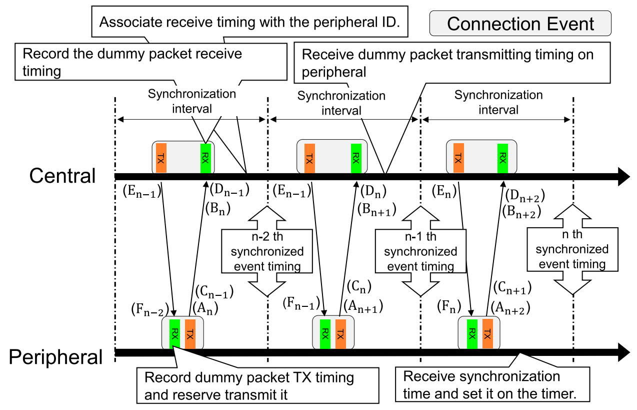

Associate receive timing with the peripheral ID.
Connection Event
Record the dummy packet receive timing
Receive dummy packet transmitting timing on peripheral
Synchronization interval
Synchronization interval
Synchronization interval
Central
Peripheral
n-2 th synchronized event timing
n-1 th synchronized event timing
n th synchronized event timing
Record dummy packet TX timing and reserve transmit it
Receive synchronization time and set it on the timer.

FIGURE 11. Communication pipelining in low-power systems.

was stopped. This timer value was sent to the Central device, which returned the synchronization timing to the Peripheral. Because the Central device calculated all the time as HFCLK, the Peripheral decomposed the time until the transmission timing was received from Central into the RTC and HFCLK parts, and set them in the timer. Thereafter, when transmitting dummy packets, the RTC interrupt served to warm up the HFCLK, after which the HFCLK timer started counting, and the HFCLK timer interrupt reversed the synchronous timing pin and stopped the HFCLK and HFCLK timer circuits.

C. LOW-POWER SCHEDULING

In high-accuracy scheduling, three communications are required for each synchronization. These communications are the transmission of dummy packets from the Peripheral to the Central device, the transmission timing of dummy packets, and the transmission of synchronization timing from the Central to the Peripheral. Because these communications do not conflict with each other, it is possible to reduce the power consumption by pipelining the 3-step communication and reducing the number of communications.

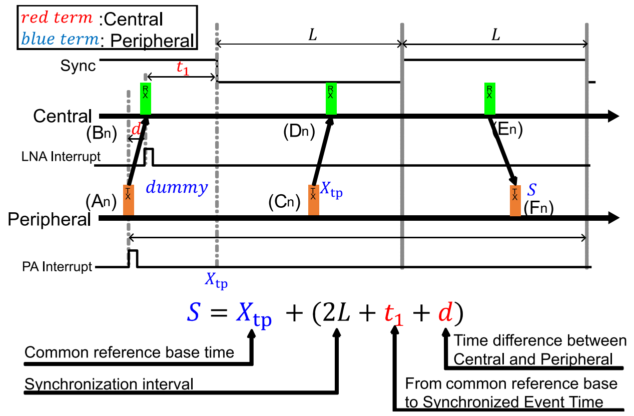


FIGURE 12. Calculations in low-power scheduling.

Figure 11 shows the concept of pipelining communication. In this figure, n represents the communication regarding the n-th synchronized event, and the figure shows three communications and synchronized events from $(n-2)$-th to n-th. First, a packet was transmitted from the Peripheral to the Central device, and the transmission timing was recorded. This packet substitutes for two functions: the n-th dummy packet and the $(n-1)$-th transmission timer value packet, as shown in Fig 6. In this communication event, the $(n-2)$-th synchronization event timing is also received and set to interrupt timer. These three functions do not conflict and occur during a single communication event.

Next, n-th dummy packet transmission timing is sent from the Peripheral to the Central device. This packet also has the function of the $(n+1)$-th dummy packet. When the n-th timing of the dummy packet transmission is received at the Central device, the synchronous event timer value on the Peripheral device is calculated from the dummy packet transmission timing value on the Peripheral device and the received timing value on the Central device.

Finally, a synchronized event-timer value is sent from the Central to the Peripheral device. The Peripheral device then sets the received timer value to that of the interrupt timer. Subsequently, a synchronous event is generated.

This scheduling method have two main advantages. First, the number of communications is less than that in high-accuracy scheduling. It consumes less power than the high-accuracy scheduling. Second, this method does not require Slave Latency. In a multi-Peripheral environment with Slave Latency, the communication order may change, which causes power overhead. This scheduling does not require Slave Latency and the communication order is stabilized. The weakness of this method is that it requires two or more synchronization intervals for synchronous scheduling, which can cause accumulation of timer drift errors.

Fig. 12 shows the calculation of the synchronization timing in low-power scheduling. For simplicity, we have omitted the explanation of pipelining and described a single-synchronization calculation. Three communications were required for each synchronization. Subsequently, each type of communication is explained. In the first communication, a dummy packet is sent from the Peripheral to the Central device. At this time, on the Peripheral side, the transmission timing X_{tp} was recorded by the PA interrupt. Simultaneously, on the Central side, the reception time and t_1 were recorded by the LNA interrupt. Here, d was considered to be a constant indicating the propagation delay and was determined experimentally. In the second communication, the Peripheral sends the transmission timing. Using this time as the starting point, the synchronization timing is calculated until the next time. Because the synchronization timing occurs after the third communication, the time from dummy packet reception to transmission is t_1 and the sum of $2L$ and d denotes the time lag between the PA and LNA. The sum of these times is added to X_{tp} to obtain the timer value S. In the third communication, this S is returned.

In low-power scheduling, Equation (5) defines the timer value S for the next synchronization event of the Peripheral.

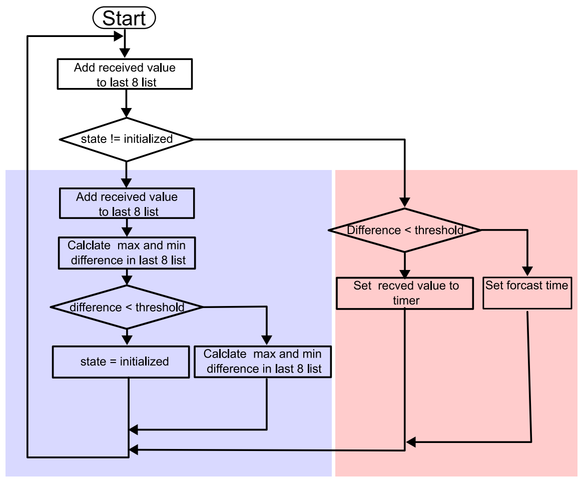
$$S = X_{\mathrm{tp}} + (2L + t_1 + d) \tag{5}$$

The primary difference from Equation (4) is the communication interval; in low-power scheduling, this interval is 1 second, which necessitates three communications and results in synchronization events lasting more than 2 seconds. X_{tp} represents common reference timing to that used in high-accuracy scheduling. The term t_1 represents the time between X_{tp} and the next synchronization event, which contrasts with its usage in high-accuracy scheduling where it has a different role. As the synchronization timing in this communication occurs $2L$ later, adding $2L$, t_1, and the error d to X_{tp} yields the next synchronization timing.

D. SIMPLE CRRECTION ALGORITHM

Packet loss during communication is unavoidable in wireless systems. To address these issues, we developed a simple correction algorithm. Fig. 13 shows a flowchart of the proposed algorithm. The algorithm is realized by combining multiple interrupt handlers, which is differs from the actual software flow. The correction algorithm has two stages: an initialization stage and a correction stage. First, in the initialization stage, the stability of the communication is determined. Given that the communication timing in BLE is not stable immediately after the connection, it is necessary to determine whether the communication is stable. The absolute difference between the maximum and minimum values of the last eight synchronization timings received from the Central was calculated; and if the difference is within the allowable value, the system is considered to be stable. The permitted value is $2\left\lceil F_{\mathrm{HFCLK}}/10^6 \right\rceil$.

The correction stage involves the use of predictive values. The correction stage uses predictive values. The predicted value is the timing of the occurrence of the Synchronized Event calculated using the value sampled from the timer on the Peripheral side of the timing of data transmission and reception. The difference between the predicted value and received Synchronized Event timing is calculated, and if this difference is less than or equal to a threshold value, then



Start
Add received value to last 8 list
state != initialized
Add received value to last 8 list
Calclate max and min difference in last 8 list
difference < threshold
state = initialized
Calclate max and min difference in last 8 list
Difference < threshold
Set recved value to timer
Set forcast time

FIGURE 13. Connection of measuring instruments.

the Synchronized Event timing received from the Central is judged to be correct. If the difference is greater than the threshold value, then the synchronization timing received from the center is determined to be correct. This value is then set to the timer of the Synchronized Event and used as the starting point for the next predicted timing. If it is greater than the threshold value, the predicted value is judged to be more accurate, and the predicted value is set in the timer. The above algorithm alleviates the outliers due to disrupted or missing communication.

V. IMPLEMENTATION AND EVALUATION

Prototyping was conducted using a commercially available Bluetooth microcontroller to demonstrate the effectiveness of the proposed method. One Central and three Peripheral devices were implemented, and actual wireless communication was performed to evaluate accuracy.

A. EXPERIMENTAL SETUP

Fig. 14 shows the devices used in the evaluation experiment. An evaluation board nRF52840DK with an nRF52840 Bluetooth microcontroller (Nordic Inc.) was used for both Central and Peripheral devices. This board has a port to connect to the Power Profiler Kit2(PPK2), shown as PPK2 in Fig. 14.

Furthermore, to measure power, LEDs and Peripheral circuits are separated from the MCU power supply, allowing only the power consumption of the MCU to be measured. The power consumption of the Peripheral node was measured using a power measurement unit (Power Profiler Kit, Nordic Inc.). PPK II is a power meter that utilizes a shunt resistor. It can measure power from 200 nA to 1 A in real time with an accuracy of 100 nA to 1 mA. In this study, power waveforms were analyzed using this device.

A logic analyzer (Digital Discovery, DIGILENT) was used to measure the synchronization error between the devices. With an 8-bit measurement target, the Digital Discovery 2 realizes high-accuracy at a sampling frequency of 800 MHz and sampling interval of 1.25 ns. Additionally, Wave Forms,

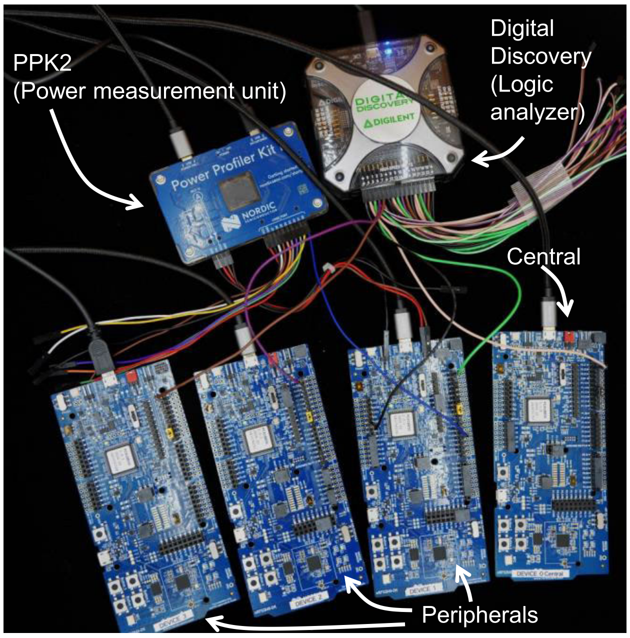


FIGURE 14. Experimental setup.

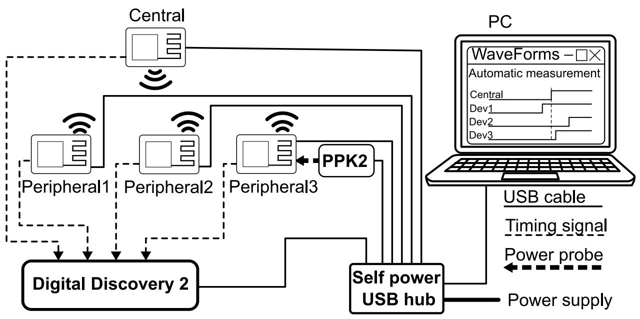


FIGURE 15. Connection of measuring instruments.

an application for viewing signal patterns obtained from Digital Discovery 2, not only displays patterns, but also has functions for data analysis and visualization using JavaScript-based scripts. In this experiment, WaveFroms function is used to trigger the rising edge of the Central timing notification pin. We then constructed an environment to measure the degree to which the increase in the Peripheral timing notification pin deviates from the Central timing notification pin.

B. EVALUATION METHOD

Fig. 15 shows a schematic of the experimental setup. The experimental system comprises of four nRF52840DKs, one of which is used as the Central unit and the remaining three are used as Peripheral units. All Peripherals are connected to the Central and synchronized to the timing of the Central synchronization timing. The Peripherals and Central have synchronization pins to notify the synchronization timing by toggling the synchronization pins. The dotted lines in the Fig. 15 connects the synchronous timing pin to the logic analyzer which detects the synchronous timing. The synchronous timing signals are connected to a Digital Discovery 2 logic analyzer. The logic analyzer is set to 800 MHz in 8-bit mode and calculates the time difference between the rise and fall timings of the Peripheral's synchronous signal relative to the Central's synchronous signal. The difference in toggle timings is automatically recorded using WaveForms as many times as required.

Fig. 16 presents a diagram illustrating the connection between the nRF52840DK and the Power Profiler Kit II. First, the power circuit output of the nRF52840DK's MCU is fed into the VIN terminal of the Power Profiler Kit II. Second, the current is measured utilizing a shunt resistor and an instrumentation amplifier. Finally, the current is routed back from the VOUT terminal back to the MCU. Peripherals, such as LEDs are connected to an independent power source, to prevent any interference with the current measurement. The entire setup, encompassing both the MCU and Peripherals, is powered by USB. The diagram has been significantly simplified to facilitate understanding, particularly the representation of the Power Profiler Kit II's power measurement circuit. For more detailed information on the circuit constants, please refer to [28].

The experimental results were compared with those of the implementation using regular timers. We will evaluate the implementation using three types of timers: RTC, high-frequency timer, and Hybrid. Additionally, the evaluation varied the frequency of high-frequency timers for synchronization algorithms involving time measurements, such as the high-frequency timer and Hybrid. The evaluation range was from 500 kHz to 16 MHz for the high-accuracy scheduling and from 500 kHz to 4 MHz for the low-power scheduling.

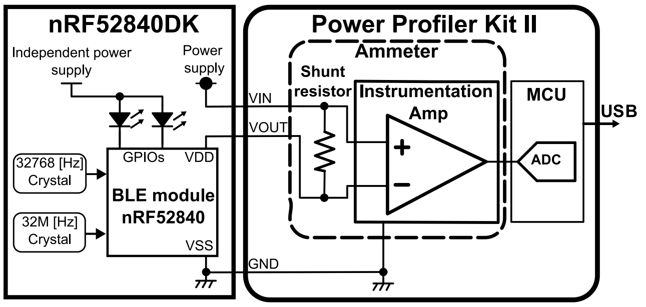


FIGURE 16. Diagram of the connection between the nRF52840DK and the Power Profiler Kit II. For details on the Power Profiler Kit II circuit, see [28].

The evaluation is based on two aspects: synchronization accuracy and power consumption. To evaluate the synchronization accuracy, three Peripherals were disconnected and reconnected five times, and 200 synchronization cycles were measured. Owing to the characteristics of BLE, the order of communication between the Central and Peripheral devices is indeterminate and may change each time the Peripherals are reconnected. The reconnection of the measurements was intended to reduce this effect. Power consumption was measured by reconnecting and taking 10 measurements of the average current over one minute, and the average of these values was considered.

C. EVALUATION RESULTS

Fig. 17 summarizes the results of each experiment. Synchronization was performed using three different timers: RTC only, HFCLK only, and the hybrid timer. The frequency of the programmable HFCLK was varied from 0.5 MHz to 16 MHz for performance evaluation. For each condition, both high-accuracy and low-power scheduling were implemented and evaluated. The hybrid timer system reduced power consumption by one order of magnitude. In addition, the synchronization accuracy was slightly improved with the hybrid method compared to the case where only HFCLK was operated. This is a slightly counterintuitive result, but it is due to the variation ranges in the frequency of the RTC and HFCLK. The frequency variation of RTC was sufficiently small compared to that of HFCLK.

Fig. 18 shows a breakdown of the relationship between the HFCLK frequency and synchronization error for a combination of scheduling schemes. In high-accuracy scheduling, there is little difference in the accuracy between the HFCLK-only and hybrid timer cases. By contrast, in low-power scheduling, the hybrid type was found to be more accurate than HFCLK. This was caused by the frequency variation between HFCLK and RTC. Low-power scheduling is strongly affected by HFCLK frequency fluctuations, which exhibit large variations because of the long periods required for synchronization. In high-accuracy scheduling, the period required for synchronization is a few hundred milliseconds, whereas in low-power scheduling, it takes a few seconds.

Fig. 19 illustrates the cumulative time synchronization error for high-accuracy scheduling at 16-MHz and low-power scheduling with a 4-MHz hybrid timer. The 99% cumulative synchronization errors are 591.25 ns and 2028.75 ns with high-accuracy and low-power scheduling, respectively.

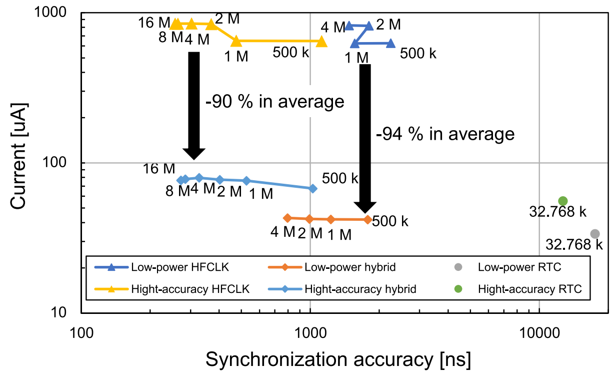


FIGURE 17. Comparison of accuracy and current for each scheduling and timer.

Fig. 20 shows an example of a current consumption waveform with a single synchronization cycle. The red and blue lines show that HFCLK is activated for dummy packet transmission and interrupts the handling for synchronization. In high-accuracy scheduling, following the transmission

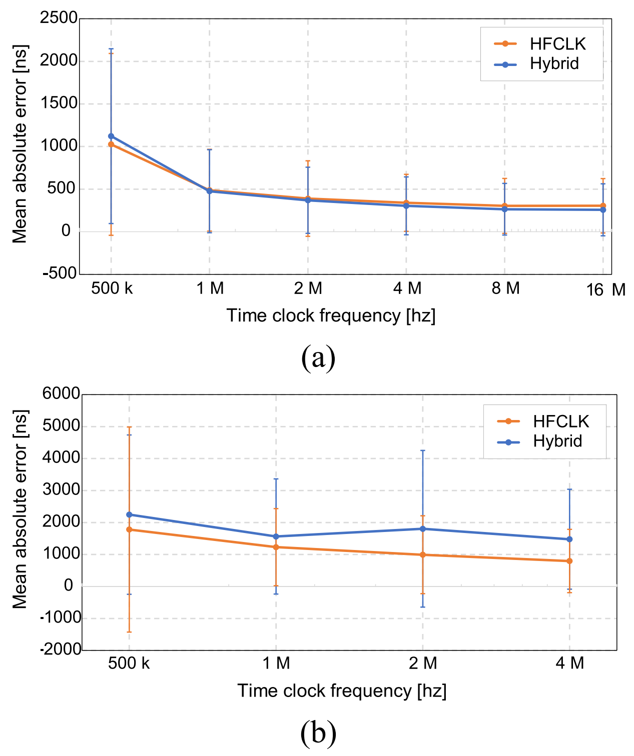


FIGURE 18. Relationship between HFCLK frequency and synchronization error with (a) high-accuracy scheduling and (b) low-power scheduling. Error bars represent standard deviations.

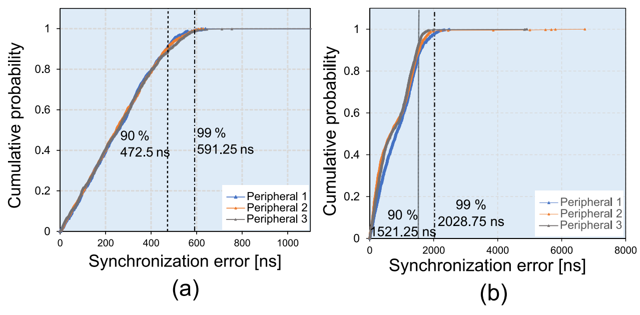


FIGURE 19. Error accumulation of (a) high-accuracy scheduling with 16-MHz HFCLK and (b) low-power scheduling with 4-MHz hybrid timer.

of dummy packets, the receiver is active multiple times in response to communications between other Peripherals and the Central device. In low-power scheduling, this power waste can be avoided by setting the Slave Latency.

As a result, the accuracy of the high-accuracy scheduling was 272 ns with a 16-MHz HFCLK, as shown in Fig. 18. The accuracy in low-power scheduling is 990 ns with a 4-MHz hybrid timer. Average power consumptions with these conditions are 252.1 μJ and 141.8 μJ, respectively. These values represent the average values measured in the same manner as the current waveforms, as shown in Fig. 20.

Tables 1 and 2 show the power reductions achieved by hybridizing timers in each scheduling method.

In high-precision scheduling, power consumption is reduced by 90%, and in low-power scheduling, it is reduced by 94%.

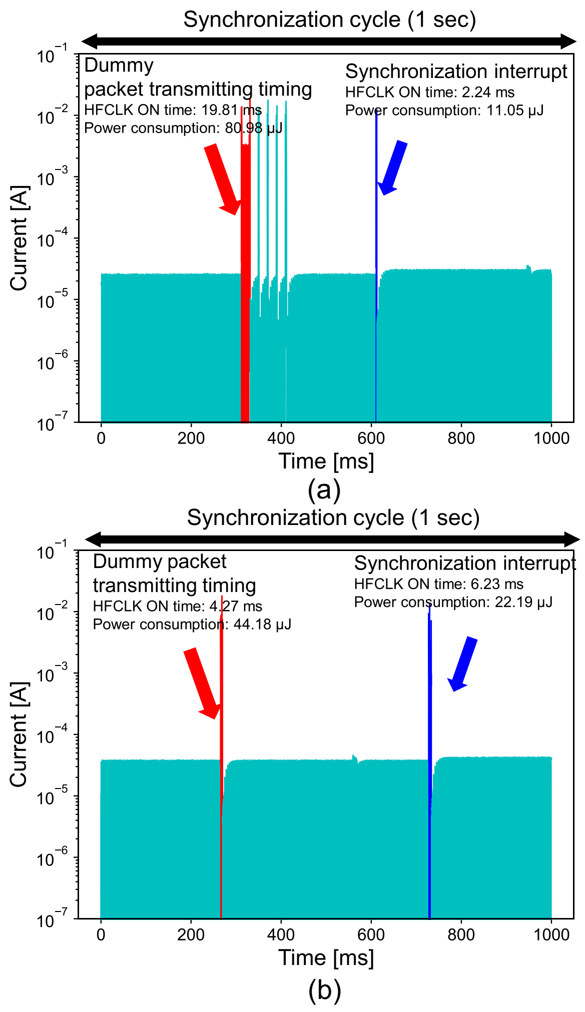


FIGURE 20. Measured current consumption waveforms while synchronization cycle with (a) high-accuracy scheduling and (b) low-power scheduling.

D. POWER ANALYSIS

To estimate the worst-case power consumption for high-accuracy scheduling and low-power scheduling, we analyzed and formulated the power consumption. Power consumption of microcontrollers supporting Bluetooth includes the CPU, radio communications, such as LNA and PA, standby power, and timers such as RTC and HFCLK.

In both scheduling modes, a current of approximately 20 μA always flows as steady-state power, P_{base}. These includes the leakage current and the power consumption of the RTC. In this section, the variables starting with P denote power consumption per second and variables starting with E denote energy consumption per event. The power consumption of high-accuracy scheduling includes three communication events: (A)-(B), (C)-(D), and (E)-(F), as well as time-synchronized events and Soft Device systems.

TABLE 1. Power reduction by timer hybridization IN high-precision scheduling.

	Timer frequency [Hz]						
Timer type	32.768k	500k	1M	2M	4M	8M	16M
HFCLK [μJ]		2134.4	2146.3	2770.4	2793.8	2791.1	2784.9
Hybrid [μJ]		222.7	250.8	225.1	262.7	257.4	252.8
(Reduction rate)		(0.10)	(0.12)	(0.09)	(0.09)	(0.09)	(0.09)
RTC	183.8						

The power consumption for the communication event between (A) and (B) includes the power used by the HFCLK timer to measure the communication timing and the power required for radio communication. The HFCLK operates from the time an interrupt occurs until the radio communication and PA pins are set. The maximum HFCLK operating time is twice Connection Interval, denoted by t_{CI}, and warm up time denoted by t_{warm}. When HFCLK timer operates at 16 MHz and it requires approximately 800 μA of current according to actual measurements. The power consumption per unit time of this timer is defined as P_{HFCLK}.

The communications (C)-(D) and (E)-(F) are communication-only events, denoted by E_{CD} and E_{EF}, respectively, and consume approximately 26 μW and 27 μW of power. Communication does not always occur during the Connection Interval and can be occasionally skipped. In such cases, the CPU remains activated to prepare for communication, consuming approximately 14 μW of power, defined as E_{sys}. The power consumption of a Synchronized Event, defined as E_{HaInt}, consumes approximately 10 μA of power. Specifically, the power consumption remains nearly constant because HFCLK's running time variation is within one RTC cycle. Based on these factors, the worst-case power consumption for high-accuracy scheduling is defined as shown in Equation (6) below.

$$P_{\text{worst}} = P_{\text{base}} + (t_{\text{warm}} + 2t_{\text{CI}}) + E_{\text{CD}} + E_{\text{EF}} + 3E_{\text{sys}} + E_{\text{HaInt}} \tag{6}$$

Next, the power for low-power scheduling is formulated. Given that low-power scheduling is conducted at regular intervals with slight uncertainty in communication, there is no requirement to target only the worst-case power consumption.

TABLE 2. Power reduction by hybridizing timers in low-power scheduling.

	Timer frequency [Hz]				
Timer type	32.768 k	500k	1M	2M	4M
HFCLK [μJ]		2066.6	2060.0	2699.2	2714.1
Hybrid [μJ]		138.3	138.9	139.6	141.8
(Reduction rate)		(0.07)	(0.07)	(0.05)	(0.05)
RTC	111.3				

Power consumption comprises of base power T_{base}, the timer and communication power denoted as E_{com}, and power consumption of synchronized events, denoted as E_{LpInt}. In E_{com}, communications between (A)and (B), (C) and (D), and (E) and (F) are pipelined and occur simultaneously. Due to the constant communication interval in low-power scheduling, it is possible to reduce the running time of the timer used for measuring communication timing. This results in a power consumption of approximately 46 μW as measured in practice. Finally, the power consumption of the synchronized event, denoted as E_{LpInt}, also remains nearly constant, consuming approximately 22 μW. Based on these factors, the power consumption for low-power scheduling can be defined as shown in Equation (7) below.

$$P_{\text{worst}} = P_{\text{base}} + E_{\text{com}} + E_{\text{LpInt}} \tag{7}$$

VI. DISCUSSION

Table 3 shows a comparison of the proposed method with previous studies on time synchronization methods using Bluetooth.

In [12], highly power-efficient time synchronization using only RTC was presented. Our high-accuracy and low-power scheduling methods achieve time synchronization that is ten times more effective than this method. The proposed method improves upon this work by achieving synchronization with less communication. In our implementation, measurements indicate that the power required for one communication is approximately 32 μJ. Based on this value, we can estimate that if the algorithm in [12] was implemented in a manner similar to our program, the estimated power overhead would be approximately 36.4% for high-accuracy scheduling and approximately -1.5% for low-power scheduling.

The largest difference between BlueSync [7] and the proposed method was the length of measurement time. Our approach is designed for synchronization at 1-second intervals, whereas BlueSync aims for longer, minute-scale synchronization and includes advanced timer drift compensation. BlueSync combines two timers, HFCLK and RTC, based on, [29] to measure the time. BlueSync and the proposed method differ in the control of the HFCLK and RTC timers. An HFCLK timer was used as the system clock, and HFCLK could not be stopped during processing. The proposed method starts not only the HFCLK timer, but also the HFCLK clock source before using the HFCLK timer and releases it after the HFCLK timer stops. Therefore, it is superior in terms of power consumption performance.

In Table 1, the communication timing is detected using comparator-based power consumption analysis [8]. The most significant difference between this method and the proposed method is the presence of an additional device to measure the current. The proposed system detects communication timing by GPIO interrupts using the Soft Device function for time synchronization. Thus, it is superior in terms of cost and power consumption because this measurement does not require Peripheral devices.

The PTP protocol is a time-synchronization protocol for an IP network. PTP can be used in the IoT and embedded systems such as PCs, servers, and sensor networks. The accuracy of this protocol is 7.8 ns [16] for MCU implementations that support PTP hardware and 15 μs for Wi-Fi MCUs that do not support PTP. Compared to time synchronization via Bluetooth, PTP has equal or better accuracy; however, the protocol is more complex and consumes more power [17]. Therefore, power-saving systems, such as those in the proposed MicroSync, will be more important for Bluetooth-based time synchronization systems.

Comparing ecoSync and MicroSync, a power-saving time synchronization method, MicroSync's low-power scheduling outperforms ecoSync's in terms of power consumption and accuracy. This is despite the fact that ecoSync's resynchronization interval is 60 min, which is 3600 times longer than MicroSync's 1-s interval. This is due to the fact that ecoSync runs on ESP32, which supports Wi-Fi, whereas our system uses nRF52840. This supports BLE, a Bluetooth power-saving standard. Additionally, the proposed system can realize higher throughput using less power by combining measurement data and time synchronization in a single packet.

The performance of MicroSync was compared with that of conventional time synchronization, which assumes acceleration measurements using an IMU. First, as shown in Table 1, MicroSync achieves 4.9 times better synchronization accuracy and 23.7 times better power consumption at a resynchronization interval of 1 s. Second, we compared and analyzed the conditions from both hardware and application perspectives to identify the differences. In terms of hardware, MicroSync uses nRF52840, whereas the previous study used nRF52832. Hence, there was no significant difference in power-saving performance. On the application side, MicroSync's experiment focuses only on time synchronization, whereas the previous study not only focused on synchronization but also actively acquired acceleration data. The sampling rate of the IMU is 100 Hz. However, previous methods have shown that reduced communication leads to lower power consumption; even considering that 100-Hz sampling by the IMU is insufficient to fully explain the 23.7-fold increase in power efficiency. For an accurate comparison, an analysis with a power monitor, such as PPK2, using actual equipment is necessary.

The MicroSync methodology broadens its applicability through two key technologies: timer hybridization and communication scheduling. First, most of microcontrollers with radio modules have an RTC and high frequency timers and timer hybridization are generalizable. Second, the scheduling algorithm of MicroSync is optimized for the BLE protocol and operates with minimal communications. Bluetooth is an event-driven protocol and communication events occur periodically. Therefore, scheduling is aligned with these timings. This indicates that MicroSync can be applied to periodic communication protocols. Zigbee is one such protocol. The difference from Bluetooth is that the child node queries the parent node during communication. Zigbee uses a mechanism termed as Polling for periodic communication, which makes it possible to implement low-power scheduling. Furthermore, it is possible to switch between two types of Poll intervals, Long Polling and Short Polling. This suggests the potential for implementing high-accuracy scheduling.

TABLE 3. Comparison between the proposed and conventional methods.

Method name	Data link protocol	Extra device requirement	Mean time sync. accuracy	Resync interval	Energy
Prop. (High-accuracy)	Bluetooth GATT	No	0.272 µs	1s	252.1 µW
Prop. (Low-power)	Bluetooth GATT	No	0.990 µs	1s	141.8 µW
[12]	Bluetooth GATT	No	20 µs	1s	(estimation)
BlueSync [7]	Bluetooth GAP (Advertising)	No	4.5 µs	1m	n/a
[11]	Bluetooth GATT	Power measuring circuit	0.9 µs	100 ms	n/a
PTP [16]	Ethernet	PTP supported Ethernet PHY	7.8 ns	Continuous	n/a
PTP (Software) [17]	Wi-Fi	No	15 µs	Continuous	386 mW
PTP [19]	Bluetooth GATT	FPGA	15 ns	Continuous	n/a
ecoSync [17]	WiFi	No	42 µs	60 min	182.83 µW
[22]	Bluetooth GATT	No	4.4 µs	1s	3.34 mW
[22]	Bluetooth GATT	No	200 µs	60 s	55 µW

The following is a summary of the points on which our method outperforms conventional methods. Firstly, the power consumption of timers is a critical issue in power-saving time synchronization systems, where timers with high frequency and high resolution typically lead to increased power consumption. The proposed method uses a hybrid timer to achieve both high time synchronization accuracy and time resolution. Secondly, in time synchronization, shorter synchronization intervals lead to more frequent communication, and consequently, greater power consumption. However, extending the communication interval often results in decreased accuracy due to timer drift. Conversely, MicroSync achieves short resynchronization cycles and power savings by optimizing the hybrid timer and communication scheduling. This capability suggests that when MicroSync is employed in a sensor network, it enables high-frequency data sampling with both high-accuracy and low power consumption.

Based on the experimental results and the characteristics of BLE, we predicted the practical limitations of the MicroSync application. Firstly, the Soft Device specification limits the number of connectable Peripherals to a maximum of 20 devices. Consequently, MicroSync can only connect to 20 Peripherals per central device. Secondly, if multiple Peripherals request quick transmissions at random times, the communication timing may become irregular, potentially leading to unstable operations. Therefore, the maximum number of Peripherals that can stably operate is likely to be even lower. Thirdly, this study focused only on time synchronization. If MicroSync is applied to a sensor system that gathers data intensively from Peripherals, then the power consumption may escalate as the volume of data communication increases.

In MicroSync, we observed time synchronization errors of 272 ns in high-accuracy scheduling and 990 ns in power-saving scheduling. To investigate the causes, we examined the communication timing and timer cycles on the Central and Peripheral devices. The investigation revealed that the timer values between the Central and Peripheral devices occasionally exhibited discrepancies ranging from several tens to approximately 100 counts at a frequency of 16 MHz. Potential factors influencing these dis Using MicroSync, we observed time synchronization errors of 272 ns for high-accuracy scheduling and 990 ns for power-saving scheduling. To investigate the causes, we examined the communication timing and timer cycles on both the Central and devices. The investigation revealed that the timer values between the Central and Peripheral devices occasionally exhibited discrepancies ranging from several tens to approximately 100 counts at a frequency of 16 MHz. Potential factors influencing these discrepancies are considered as follows.

- Communication Detection Error: MicroSync utilizes the notification features of LNA and PA in SoftDevice to detect communication timings. When LNA or PA is activated, SoftDevice raises a designated pin to signal an event. In our system, this change is detected through a GPIO interrupt, which enables time synchronization. Therefore, the timing of notification pin activation and the duration of GPIO interrupts introduce uncertainties in synchronization.
- Disorders in Communication Order: In BLE, multiple Peripherals are connected to a single Central unit and communicate in a time-divided, concurrent processing manner. Therefore, the order of Peripherals can change during communication, leading to disturbances in communication timing.
- Communication Delays: In wireless systems, various factors, such as the transmission time in the physical layer, the propagation time of information, and the reception time, can cause delays. In the application layer, these delays manifest as the difference between the rise times of the PA pin on the transmitter side and LNA pin on the receiver side. Disturbances in these delay times due to interference with other devices can adversely affect synchronization accuracy.
- Recovery Time from Sleep: To enhance power efficiency, our system places nRF52840 into deep sleep. The recovery time from sleep in nRF52840 has two delay modes, and our system selects a more power-efficient mode. This introduces uncertainties in the time it takes to wake up from a timer interruption.

- Timer Drift: MicroSync assumes synchronization intervals of up to one second, aligning with the maximum Connection Interval of one second due to the characteristics of the nRF52840. Thus, clock drift does not dominate synchronization errors, and we did not
- implement any correction for clock drift.

VII. CONCLUSION

We focused on the power consumption of Bluetooth time synchronization and proposed MicroSync, which is a synchronization method that achieves sub-microsecond accuracy with power requirement of tens of microamperes. MicroSync is based on two elements: timer hybridization and communication scheduling. Two types of MicroSync implementations are proposed and experimentally evaluated: high-accuracy scheduling and low-power scheduling. The proposed method is implemented and evaluated using a commercially available Bluetooth microcontroller. The evaluation results show that the proposed method achieves time synchronization with an accuracy of less than 1 μs. This is a higher accuracy than conventional methods, and hybridization has shown that synchronous processes can be realized with very low energy consumption.

These results show that the accuracy overhead due to timer hybridization is negligible. Furthermore, the results of the power performance comparison show that the synchronization system with 1-s intervals saves the most power. These results show that the two hypotheses presented in the Introduction are correct and useful.

REFERENCES

[1] T. Polonelli, A. Moallemi, W. Kong, H. Müller, J. Deparday, M. Magno, and L. Benini, "A self-sustainable and micro-second time synchronized multi-node wireless system for aerodynamic monitoring on wind turbines," *IEEE Access*, vol. 11, pp. 119506–119522, 2023, doi: 10.1109/ACCESS.2023.3327422.

[2] I. Toru, Y. Yasuda, S. Sato, S. Izumi, and H. Kawaguchi, "Millimeter-precision ultrasonic DSSS positioning technique with geometric triangle constraint," *IEEE Sensors J.*, vol. 22, no. 16, pp. 16202–16211, Aug. 2022, doi: 10.1109/JSEN.2022.3188007.

[3] S. Sato, Y. Yasuda, R. Ohara, R. Hamabe, T. Genda, S. Imanaka, S. Izumi, and H. Kawaguchi, "Estimating person location in bathroom with spatial ultrasound and variational autoencoder," in *Proc. IEEE Int. Ultrason. Symp. (IUS)*, Sep. 2023, pp. 1–26.

[4] L. D. Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014, doi: 10.1109/TII.2014.2300753.

[5] R. Heydon, *Bluetooth Low Energy? The Developer's Handbook*. New York, NY, USA: Prentice-Hall, 2012.

[6] F. J. Dian, A. Yousefi, and K. Somaratne, "Performance evaluation of time synchronization using current consumption pattern of BLE devices," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2018, pp. 906–910, doi: 10.1109/CCWC.2018.8301666.

[7] F. Asgarian and K. Najafi, "BlueSync: Time synchronization in Bluetooth low energy with energy-efficient calculations," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8633–8645, Jun. 2022, doi: 10.1109/JIOT.2021.3116921.

[8] K. Somaratne, F. J. Dian, and A. Yousefi, "Accuracy analysis of time synchronization using current consumption pattern of BLE devices," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2018, pp. 841–844, doi: 10.1109/CCWC.2018.8301624.

[9] A. Yousefi, K. Somaratne, and F. J. Dian, "Analysis of time synchronization based on current measurement for Bluetooth low energy (BLE)," in *Proc. 8th IEEE Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Oct. 2017, pp. 602–607, doi: 10.1109/IEMCON.2017.8117157.

[10] S. Sridhar, P. Misra, G. S. Gill, and J. Warrior, "Cheepsync: A time synchronization service for resource constrained Bluetooth le advertisers," *IEEE Commun. Mag.*, vol. 54, no. 1, pp. 136–143, Jan. 2016, doi: 10.1109/MCOM.2016.7378439.

[11] C. C. Rheinländer and N. Wehn, "Precise synchronization time stamp generation for Bluetooth low energy," in *Proc. IEEE Sensors*, Oct. 2016, pp. 1–3, doi: 10.1109/ICSENS.2016.7808812.

[12] M. Harada, S. Izumi, R. Kozeni, Y. Yoshikawa, T. Ishii, H. Kawaguchi, S. Uemura, and K. Araki, "20-μs accuracy time-synchronization method using Bluetooth low energy for Internet-of-Things sensors," in *Proc. IEEE 19th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2022, pp. 181–186, doi: 10.1109/CCNC49033.2022.9700687.

[13] J. Li, E. Quintin, H. Wang, B. E. McDonald, T. R. Farrell, X. Huang, and E. A. Clancy, "Application-layer time synchronization and data alignment method for multichannel biosignal sensors using BLE protocol," *Sensors*, vol. 23, no. 8, p. 3954, Apr. 2023.

[14] P. Enge and P. Misra, "Special issue on global positioning system," *Proc. IEEE*, vol. 87, no. 1, pp. 3–15, Jan. 1999.

[15] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, Standard IEEE Std 1588-2008, 1588, pp. 1–499.

[16] S.-Y. Kim, D.-O. Chung, K. Lee, C. Lee, and J. Huh, "Low-power always-on camera (AoC) system with workload offloading to CMOS image sensor," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2023, pp. 1–24.

[17] S. Puckett and E. Jovanov, "EcoSync: An energy-efficient clock discipline data synchronization in Wi-Fi IoMT systems," *Electronics*, vol. 12, no. 20, p. 4226, Oct. 2023.

[18] P. Chen and Z. Yang, "Understanding precision time protocol in todays Wi-Fi networks: A measurement study," in *Proc. USENIX Annu. Tech. Conf.*, Jul. 2021, pp. 597–610.

[19] A. S. Nagra, M. A. Pasha, and S. Masud, "FPGA implementation of IEEE 1588 protocol for Bluetooth-based distributed wireless systems," in *Proc. 29th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Oct. 2022, pp. 1–4, doi: 10.1109/ICECS202256217.2022.9970850.

[20] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Operating Syst. Rev.*, vol. 36, no. SI, pp. 147–163, Dec. 2002, doi: 10.1145/844128.844143.

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

[22] J. Cappelle, S. Goossens, L. De Strycker, and L. Van der Perre, "Low-power synchronization for multi-IMU WSNs," *IEEE Embedded Syst. Lett.*, vol. 16, no. 2, pp. 210–213, Jun. 2024.

[23] M. Elsharief, M. A. Abd El-Gawad, and H. Kim, "Low-power scheduling for time synchronization protocols in a wireless sensor networks," *IEEE Sensors Lett.*, vol. 3, no. 4, pp. 1–4, Apr. 2019, doi: 10.1109/LSENS.2019.2902389.

[24] M. Elsharief, M. A. Abd El-Gawad, and H. Kim, "FADS: Fast scheduling and accurate drift compensation for time synchronization of wireless sensor networks," *IEEE Access*, vol. 6, pp. 65507–65520, 2018, doi: 10.1109/ACCESS.2018.2878272.

[25] M. Elsharief, M. A. A. El-Gawad, H. Ko, and S. Pack, "LPSRS: Low-power multi-hop synchronization based on reference node scheduling for Internet of Things," *Energies*, vol. 15, no. 6, p. 2289, Mar. 2022.

[26] M. Elsharief, A. A. Emran, H. Hassan, S. R. Sabuj, and H.-S. Jo, "SLES: Scheduling-based low energy synchronization for industrial Internet of Things," *IEEE Sensors J.*, vol. 22, no. 16, pp. 16652–16661, Aug. 2022, doi: 10.1109/JSEN.2022.3188758.

[27] M. Elsharief, A. Emran, H. Hassan, S. R. Sabuj, and H.-S. Jo, "Energy-efficient synchronization in industrial Internet of Things: An intelligent neighbor-knowledge approach," *IEEE Trans. Ind. Informat.*, vol. 20, no. 6, pp. 8548–8558, Jun. 2024, doi: 10.1109/TII.2024.3369236.

[28] (2024). *Power Profiler Kit II—Downloads—Nordicsemi.com*. [Online]. Available: https://www.nordicsemi.com/Products/Development-hardware/Power-Profiler-Kit-2/Download

[29] T. Schmid, P. Dutta, and M. B. Srivastava, "High-resolution, low-power time synchronization an oxymoron no more," in *Proc. 9th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2010, pp. 151–161, doi: 10.1145/1791212.1791231.

RYOTARO OHARA received the Foundation degree from Tokyo Metropolitan College of Industrial Technology, in 2016, the B.Eng. degree in environmental and life science from Toyohashi University, in 2018, and the master's degree in science, technology and innovation from Kobe University, Hyogo, Japan, in 2021. His current research interest includes 3D environment monitoring by ultrasonic.

SHINTARO IZUMI (Member, IEEE) received the B.Eng. and M.Eng. degrees in computer science and systems engineering and the Ph.D. degree in engineering from Kobe University, Hyogo, Japan, in 2007, 2008, and 2011, respectively.

He was a JSPS Research Fellow at Kobe University, from 2009 to 2011; an Assistant Professor with the Organization of Advanced Science and Technology, Kobe University, from 2011 to 2018; and an Associate Professor with the Institute of Scientific and Industrial Research, Osaka University, from 2018 to 2019. Since 2019, he has been an Associate Professor with the Graduate School of System Informatics, Kobe University. His current research interests include biomedical signal processing, communication protocols, low-power VLSI design, and sensor networks. He has served as a Technical Committee Member for the IEEE Biomedical and Life Science Circuits and Systems, a Student Activity Committee Member for the IEEE Kansai Section, and a Program Committee Member for the IEEE Symposium on Low-Power and High-Speed Chips (COOL Chips). He was a recipient of the 2010 IEEE SSCS Japan Chapter Young Researchers Award. He was the Chair of the IEEE Kansai Section Young Professionals Affinity Group.

SHOYA IMANAKA (Graduate Student Member, IEEE) received the B.Eng. degree in computer science and systems engineering from Kobe University, Hyogo, Japan, in 2023. He is currently pursuing the M.Inf. degree with the Graduate School of Informatics, Kyoto University, Kyoto, Japan. His research interests include the IoT sensor networks, the IoT monitoring service, and multi-access edge computing.

TETSUO YAMAMURA received the B.Eng. degree in communication engineering from the Department of Engineering, Tokai University, Kanagawa, Japan, in 1984, and the master's degree in electrical engineering from the Graduate School of Engineering, Tokai University, in 1986.

In 1986, he joined Matsushita Electric Industrial Company Ltd. (now Panasonic Connect Company Ltd.), where he was actively involved in the design and development of personal computers, until 2020. He is currently serving as an Academic Researcher with the Graduate School of Science, Technology and Innovation, Kobe University, Hyogo, Japan. His research interests include the development of sensing technologies for the realization of IoT society.

ISHII TORU received the B.Eng. degree in electronic engineering from Kyoto University, Kyoto, Japan, in 1986, and the M.B.A. and Ph.D. degrees in science, technology, and innovation degree from Kobe University, Kobe, Japan, in 2013 and 2022, respectively. In 1986, he joined Minolta, Osaka, Japan, where he developed digital imaging devices. In 1999, he moved to Murata Manufacturing, Kyoto, where he was the Development Manager for automotive radars, communication modules, and sensors. Since 2018, he has been engaged in research on ultrasound sensing at Kobe University.

HIROSHI KAWAGUCHI (Member, IEEE) received the B.Eng. and M.Eng. degrees in electronic engineering from Chiba University, Chiba, Japan, in 1991 and 1993, respectively, and the Ph.D. degree in electronics engineering from The University of Tokyo, Tokyo, Japan, in 2006. In 1993, he joined the Konami Corporation, Kobe, Japan, where he developed arcade entertainment systems. He moved to the Institute of Industrial Science, The University of Tokyo, as a Technical Associate, in 1996, and was appointed as a Research Associate, in 2003. In 2005, he moved to the Graduate School of Engineering, Kobe University, Kobe, as a Research Associate. From 2015 to 2016, he was a Visiting Researcher with Politecnico di Milano. Since 2016, he has been a Full Professor at the Graduate School of Science, Technology and Innovation, Kobe University. He is also a Collaborative Researcher with the Institute of Industrial Science, The University of Tokyo. His current research interests include low-voltage operating circuits, soft error characterization and mitigation, ubiquitous sensor networks, organic semiconductor circuits, data converters, healthcare devices, and neuro computer architecture. He is a member of ACM and IEICE. He was a recipient of the IEEE ISSCC 2004 Takuo Sugano Outstanding Paper Award, the ACM/IEEE ASP-DAC 2013 University Design Contest Best Design Award, and the IEEE ICECS 2016 Best Paper Award. He has served as a Design and Implementation of Signal Processing Systems (DISPS) Technical Committee Member for the IEEE Signal Processing Society; a Technical Program Committee Member for IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE Global Conference on Signal and Information Processing (GlobalSIP), IEEE Workshop on Signal Processing Systems (SiPS), IEEE Custom Integrated Circuits Conference (CICC), and IEEE Symposium on Low-Power and High-Speed Chips (COOL Chips); and an Organizing Committee Member for IEEE Asian Solid-State Circuits Conference (A-SSCC) and ACM/IEEE Asia and South Pacific Design Automation Conference (ASP-DAC). He was an Associate Editor of *Journal of Signal Processing Systems* (Springer), *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *IEICE Transactions on Electronics*, and *IPSJ Transactions on System LSI Design Methodology* (TSLDM).