Annals of Telecommunications (2022) 77:221–236
https://doi.org/10.1007/s12243-021-00877-5

A novel offline indoor acoustic synchronization protocol: experimental analysis

Zahi Nakad[1] · Mohammad Ali Sayed[1] · Anthony Yaghi[1] · Harag Margossian[1] · Wissam Fawaz[1]

Received: 8 October 2020 / Accepted: 9 August 2021 / Published online: 18 August 2021

Abstract
Smart electronic devices are playing a fundamental role in modern home and industrial applications. The increased reliance on such devices, especially in time critical and secure applications, intensifies the need for time synchronization among multiple devices. This work presents a novel audio-based, cheap, offline synchronization method, whereby multiple slaves synchronize simultaneously to a master within a single room. Synchronization is carried out under the proposed protocol in a way that is independent of the physical location of the target devices, which in turn are not required to have any sort of network connectivity. The proposed method relies on the transmission of a De Bruijn sequence that holds the information required for the slaves to synchronize. The effectiveness of the proposed synchronization protocol is validated through an in-house experimental setup. Synchronization at distances of up to 250 cm between the master and a slave was achieved.

Keywords Audio signals · De Bruijn sequence · Synchronization

1 Introduction

As smart electronic devices are becoming more abundant, we are becoming more dependent on their operation in the different aspects of our lives. As a direct result, their accurate timing is becoming a more important requirement than ever before. Synchronization is required for applications ranging from small-scale smart homes to the operation of the power system. In [1], the authors discuss the synchronization of home appliances and devices connected to a single home-area network. The need to synchronize indoor networks of sensors to govern the efficient operation of an HVAC (heating, ventilation, and air conditioning) system is presented in [2]. In [3] and [4], the importance of properly synchronized measurements for the operation of the power system is highlighted. Without proper time stamps, the system state estimated based on these measurements can be incorrect leading to misguided and sub-optimal decision making. This in turn can reduce the reliability and/or efficiency of the entire system. Synchronization in power systems is needed not only for wide area network measurements but also for components within a single control room/substation.

Several synchronization schemes are available to ensure that multiple devices are synchronized to a common time reference or a master clock. These methods range from using an Internet connection [1, 5–7], a Global Position System (GPS) timestamp [3, 8–10], repetitive pulses of FM (frequency modulation) signals [11], radio frequency (RF) signals [12], and light [13]. However, each of these schemes has its drawbacks that this work attempts to address. In network-connected applications, there is always the possibility that the Internet connection is lost or subject to a delay or denial of service (DoS) attack [14]. A GPS signal has been proven to be vulnerable to spoofing [15–17]; the authors in [18] even proposed a method for the development of a portable civilian GPS spoofer. Although the reliability of RF signals has been proven and there is an abundance of standards governing their operation, the needed hardware for using this type of communication might not always be available

✉ Wissam Fawaz
wissam.fawaz@lau.edu.lb

Zahi Nakad
zahi.nakad@lau.edu.lb

Mohammad Ali Sayed
mohammadali.sayed@lau.edu

Anthony Yaghi
anthony.yaghi@lau.edu

Harag Margossian
harag.margossian@lau.edu.lb

[1] Lebanese American University, Byblos, Lebanon

on preexisting devices requiring synchronization. When it comes to the light-based and camera approaches, the operation depends on a direct line of sight (LOS) which stands firmly in the way of concurrent practical synchronization of multiple devices.

In light of the above discussion, the objective of this study is the creation of a wireless broadcast synchronization scheme that is accurate, simple to implement, energy efficient, offline, cheap, and that uses basic ubiquitous hardware components. This is achieved by the use of audio signals as a medium for synchronization. One of the main requirements underlying the operation of the proposed system is the ability to concurrently synchronize multiple smart devices present in the same room. The first reason behind using sound is the availability of the required hardware on most smart devices. The second reason lies in its ability to enable devices to communicate together without being connected to a common network. Finally, LOS is not required as this scheme takes advantage of the broadcast nature of sound. The authors of [19] address the use of inaudible sound for indoor synchronization purposes. They focus on a method that can mitigate the long time required for estimating the starting phase in using De Bruijn sequence by utilizing a table-lookup method instead of correlation. To the best of the authors' knowledge there are no other research work that addresses using De Bruijn sequence for synchronization using acoustic signals.

Audio signals have been used to transmit data for commercial purposes by several companies. One such example is Chirp [20]. Chirp uses sound waves to transmit data between different devices which are only required to be equipped with a microphone and speaker. Acoustic signals have been utilized to classify events in surveillance systems [21]. Sound has also been used for underwater communication and synchronization purposes and was the subject of multiple studies. As an example, the authors of [22] provide a review of biologically inspired covert underwater acoustic communication, and the work presented in [23] provides a routing metric for multi-hop underwater acoustic metrics. On the other hand, the work presented in [24] and [25] aims at synchronizing multiple devices referred to as nodes. The authors of [24] present a method that is based on pairwise communication between nodes to overcome clock drift and keep the multiple slave nodes synchronized to a master node. In contrast, the method proposed in [25] takes advantage of the broadcast nature of sound for more efficient communication. These nodes are synchronized to several devices in the system having accurate time stamps. Both [24] and [25] will be further discussed in Section II.

This work utilizes the Universal Asynchronous Receiver Transmitter (UART) transceivers of two simple processors (PIC 18F4550), one to broadcast the synchronization signal and the other to receive it and synchronize the receiver's clock accordingly. The strength of this scheme is that it can be implemented on preexisting processors of almost any smart device without the need for extra processing power. The implementation of this scheme only requires UART transceivers with a microphone and a signal conditioning circuit. In [13], the authors utilized a De Bruijn sequence [26, 27] to help in achieving energy efficiency by requiring that the receiver only wake up to a portion of the synchronization signal and still synchronize correctly, as discussed further in Section II. This is a direct consequence of the De Bruijn sequence property stipulating that any specific n-bit window in such a sequence is found at a unique location within a 2n bit sequence [26, 27]. For this reason, the De Bruijn sequence was adopted in this work to send synchronization data.

The next section gives a summary of the related studies. Section III describes the system design. Section IV reports the experimental results. Finally, Section V concludes the paper.

2 Background

A raft of studies has considered the topic of time synchronization. The work in [7] described the effect of clock drift problems on applications requiring common timing. The authors addressed the importance of overcoming this clock drift via the Network Time Protocol (NTP). This would ensure that all devices connected to the Internet will maintain the same accurate timing.

The work in [6] demonstrates through an experiment the ability of Precision Time Protocol (PTP) to synchronize phasor measurement units (PMUs) in a power system. This is done by synchronizing a GrandMaster clock (GM) to UTC via a GPS receiver and then connecting its output to the PMU's clock. The latter exchanges IEEE 1588 synchronization messages with the GM to reconstruct the exact time reference. However, PTP is vulnerable to desynchronization attacks as described in [14]. In addition, synchronization attacks can be used to manipulate phase angle measurements and act as false data injection attacks against state estimation as described in [28].

The authors of [8] tested the possibility of synchronizing multiple standalone devices to the time stamp of a GPS signal. Note that GPS signals are mainly used to provide accurate positioning but have extremely precise time stamps as a byproduct. This byproduct is due to the precise timing requirements of satellites needed for proper operation. The authors described how to utilize the data available in a GPS signal by extracting the exact, stable, and common time stamp to be used for correcting the drift of internal clocks in the considered devices. The observed results showed that two devices were able to be

synchronized to the exact UTC timing with a mean relative accuracy of 2.9 ns in a 24-h period and − 1.3 ns in a 1-week time period. It is also important to note that the peak error was observed in the 1-week scenario and found to be equal to 550 ns. In both [6] and [8], GPS signals were successfully used to synchronize the clocks of various elements. This work did not consider the use of GPS signals for synchronization due to the fact that they are not available indoors and that most simple smart devices do not have GPS modules. In addition, GPS signals are known to be vulnerable to spoofing [15–18].

In [11], the authors proposed a synchronization scheme based on FM radio. A new FM receiver was designed with the ability to extract a periodic pulse from FM broadcasts, referred to as the RDS (Radio Data System) clock. This periodic pulse was then used as the message to which devices are synchronized. This scheme was tested in a lab for a period of 6 days as well as in a vehicle moving in a metropolitan area of 40 km^2. The experiment showed that this scheme is stable and hence is a viable means for calibrating the clocks of large-scale sensor networks.

The scheme also predicts the drift error and calibrates the different device clocks via the RDS clock. The results showed that this scheme was able to achieve accurate and precise clock synchronization across the different devices. The main disadvantage of this synchronization scheme is the need for an FM receiver on each smart device.

The authors of [24] provided a scenario with multiple underwater nodes communicating with one another using sound waves. These nodes are distributed such that each node is in communication with a maximum of six geographically close nodes and synchronized to a master node called node zero. In this communication scheme, a sending node is targeting a specific receiver while a receiver node can receive data from multiple nodes. A time multiplexing scheme was used to overcome interference between data packets. Dead time between time slots is introduced to guarantee correct reception. This dead time is equal to the transmission delay and an extra guard time so that the echoes of the previous packet would die out before another packet is expected to be received. Node zero begins a pairwise synchronization scheme by sending an initialization packet to node one and records the time it sent this message. When node one receives this packet, it sends back a synchronization packet of its own with the receive time in the header. Node zero takes note of the time and calculates the delay between these two nodes. It then sends back a packet with these time stamps to node one so that it would calculate the delay as well. This process is repeated between every pair of nodes that can communicate with one another. The authors of [24] were able to synchronize and communicate data between multiple nodes using sound waves while taking into consideration the effect of echoes and interference.

The authors of [25] also describe an underwater communication scheme using sound waves between multiple active nodes. This scheme defines upper hop and lower hop directions for communication and also differentiates between beacon nodes that have exact time stamps and regular nodes that relay information. When a new beacon is introduced, it broadcasts a notification with hop number 1 and its own ID. A regular node that receives this notification adds this beacon's ID to its lower hop neighbor set. The regular node then increments its hop number with respect to this beacon and broadcasts its ID with the its updated hop number relative to that specific beacon. Any regular node receiving this broadcast updates its hop distance with respect to the beacon. This way a node knows its own hop distance with respect to a specific beacon as well as the hop information of all its neighbors from that beacon. Whenever a regular node wants to synchronize, it broadcasts a request asking for synchronization from its lower hop neighbors. It records the time it sent this request and the time it received an answer as well as the send and receive time stamps of the neighbors answering this request. This pairwise synchronization is iterated for added accuracy. The original request is then transmitted all the way down the hop chain to the beacon and back from the beacon to the node that sent it. Due to the broadcast nature of the requests and replies, the node will have time stamp data from all its lower hop neighbors originating from several beacons. In summary, the proposed scheme in [25] was successful in synchronizing scattered nodes using audio signals underwater.

The work in [26] proved the existence of unique sets of periodic sequences later referred to in the literature as De Bruijn sequences. The main property of these sequences is that any n consecutive digits can be found uniquely at one and only one location within a 2n cycle. The background on the applications of the De Bruijn sequence is quite extensive, the following attempts to highlight some of the application areas that have used this sequence. In communication systems, for example, the work in [29] investigated the use of the De Bruijn sequence as user spreading codes in DS-CDMA systems; while the authors of [30] propose using an even–odd equivalent (EOE) pseudo De Bruijn sequence for primary synchronization of private mobile radio (PMR) over LTE to increase security. In [31], the researchers investigate the use of constrained De Bruijn sequence in correlation with racetrack memories to correct synchronization errors such as deletions and sticky insertions. Meteroid trajectories can be recorded by using long-exposure fireball photographs. Then by using periodic shuttering, the meteoroid velocity can be determined. The researchers in [32] prove that by using a De Bruijn sequence to drive the periodic shutter, they were able to eliminate the need of a separate subsystem to record absolute fireball timing and were able to reach sub-millisecond resolution. The work in [13] proposes

a synchronization scheme that uses LEDs and visible light as the medium for signal transmission and that targets energy-constrained devices. Under this scheme, the receiver is only required to be awake for a portion of the synchronization signal and then goes back to sleep. The receiver then wakes up before the synchronization point is reached and performs heavy sampling to obtain the exact synchronization point. This large amount of data was then processed with the help of an online algorithm. The De Bruijn sequence was used as the synchronization signal. The list of applications of this sequence is quite extensive and covers expansive areas of research, the provided examples only skim the surface. Our specific application of using this sequence for clock synchronization is closest to the manner used in [13]. The receiver in our approach does wake up periodically to catch the synchronization sequence, but the synchronization is done by utilizing the timestamps of the received packets and synchronizing back to the initial point of the sequence, which differs significantly from the technique presented in [13]. The visible light scheme used in [13] was not utilized due to its reliance on LOS that greatly limits its feasibility to synchronize multiple components in one room.

In light of the above discussion, our proposed scheme utilizes sound wave communication and a De Bruijn sequence to achieve synchronization in an indoor scenario. The following section focuses on the novelty of the design. The main contribution of this work can be summarized as follows:

1) The development of a novel cost-effective offline acoustic-based synchronization scheme.
2) The implementation of a hardware prototype with a view to verifying the validity and analyzing the performance of the proposed scheme.
3) Experimental investigation of the effects of different microphone setups (using insulation, metal plate) on the directivity and reach of the acoustic synchronization scheme.

The proposed scheme can be the primary synchronization mechanism in rooms that are not inhabited. Alternatively, it can be used as a backup offline system that does not run continuously, in which case its main use would be to handle any error or detected cyber-attack compromising the primary synchronization mechanism.

3 System design

The proposed synchronization system operates in a master–slave mode. In particular, the master has the responsibility of acoustically transmitting the De Bruijn-based synchronization sequence to the slave devices; which in turn utilize the received sequence to adjust their clocks. The master and slaves are assumed to be located within a room.

3.1 Proposed protocol

The master sends the synchronization sequence once every T_{p_Master} time units and the whole sequence lasts T_{Sync} time units. Note that the specifics of the synchronization sequence are addressed in Section III.B. Figure 1 provides a state diagram that describes the role of the master which will be triggered at periodic intervals of length T_{p_Master} to broadcast the synchronization sequence throughout the room. The master has two modes of operation, either sending the synchronization sequence or waiting for the trigger to send the sequence. Depending on the type of application and its timing requirement, the length of the periodic intervals when the synchronization sequence should be sent and received will vary.

The slave device on the other hand is required to wake up and listen to catch the synchronization sequence, waking up periodically every T_{p_Slave} time units. For illustration purposes, if we assume that the slave device is already synchronized to the master, the clock drift is less than T_{Sync}, and $T_{p\_Master} = T_{p\_Slave}$, then the slave will always wake up while the synchronization sequence is being transmitted and as such will be able to re-align its clock with that of the master as illustrated in Fig. 2. This is the optimal operational mode that the system will always attempt to reach. Figure 3 depicts the state diagram of the slave under the various expected situations. The slave first wakes-up (*Wake – Up* state) by a trigger at T_{p_Slave} and listens to check if the synchronization sequence is being received by its signal conditioning circuit. If the sequence is being received the operation moves to the *Receive N Windows* state depicted in Fig. 3; otherwise, the slave will go into the *Wait, Active Receiver* state. While in the latter state, the slave will actively wait for the synchronization sequence. Upon the detection of the synchronization sequence, the slave transitions to the *Receive N Windows* state. The value *N*, which represents a portion of the sequence sent by the master, is chosen based on the specific parameters of the utilized De Bruijn sequence as

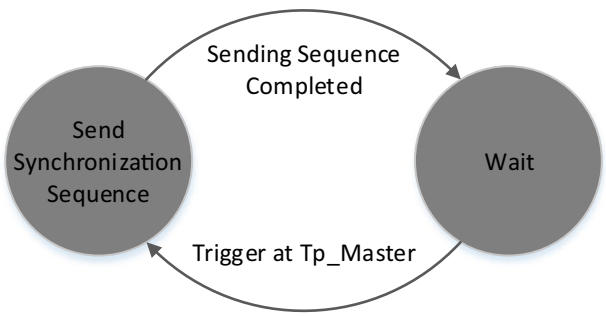

Fig. 1 State diagram for the master

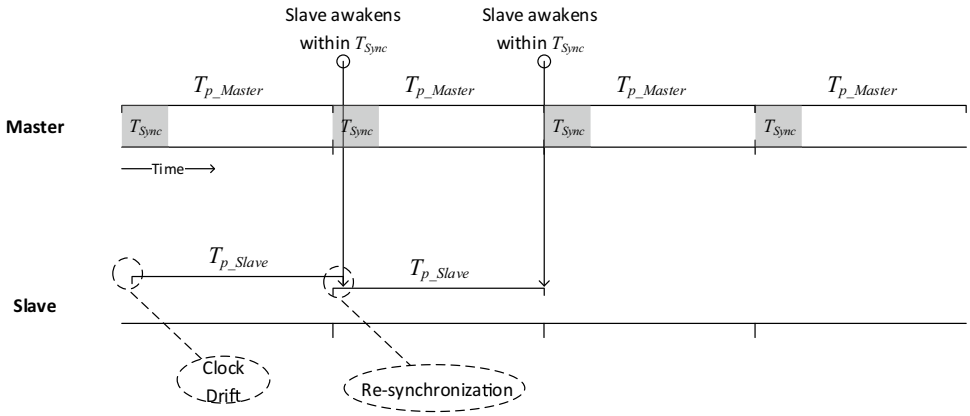

Fig. 2 Best case scenario where slaves always wake up during the transmission of the synchronization sequence and re-synchronizes with the master

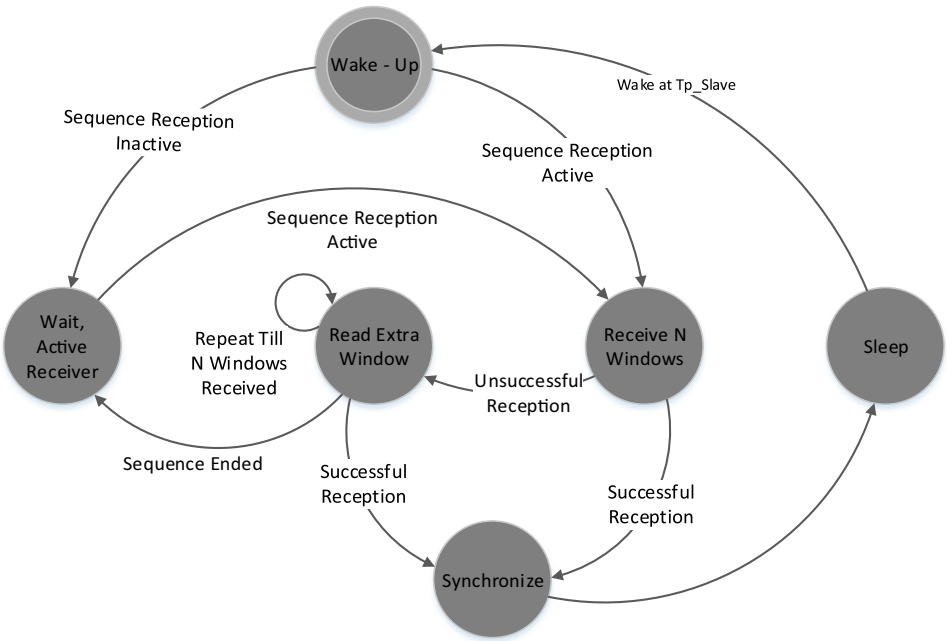

Fig. 3 State diagram for the slave

will be explained in Section III.B. On the successful reception of *N* windows, the slave moves to the *Synchronize* state where the information from the received windows is used to re-align the slave's clock with that of the master. In this case, the slave can go to the inactive low-power state *Sleep* and will wake up after a T_{p_Slave} period elapses. In case the *N* windows reception was not successful, the slave moves to the *Read Extra Window* state. In this state, it will continue receiving windows till *N* consecutive correct windows are received and the slave can thus move to the *Synchronize* state to re-adjust its clock, as explained in Section III.B. If the synchronization sequence ends before a successful reception occurs, the slave has to move to the *Wait, Active Receiver* state awaiting another sequence, representing the least optimal operation of this scheme as illustrated in Fig. 4. A detailed example is provided in the following sections to fully explain the operation of the proposed protocol.

It is worthwhile noting that T_{p_Master} and T_{p_Slave} are set according to the requirements of the specific application. For an application having stringent convergence requirements, the master can transmit the synchronization sequence continuously while slaves wake up at short intervals.

3.2 Synchronization sequence and encoding scheme

With a De Bruijn sequence, it is enough for the receiver to detect an *n-bit* window to identify the current position in the transmitted sequence. This concept is highlighted in a simplified example that follows.

Consider a *4-bit* window sequence of length 2^4 *(16 bits)* such that the sequence $Seq = 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1$ and the windows are called W_1, W_2, $W_{3\ldots}$ The first window is the first 4 bits $(W_1 = 0\ 0\ 0\ 0)$; the second window is bits 2

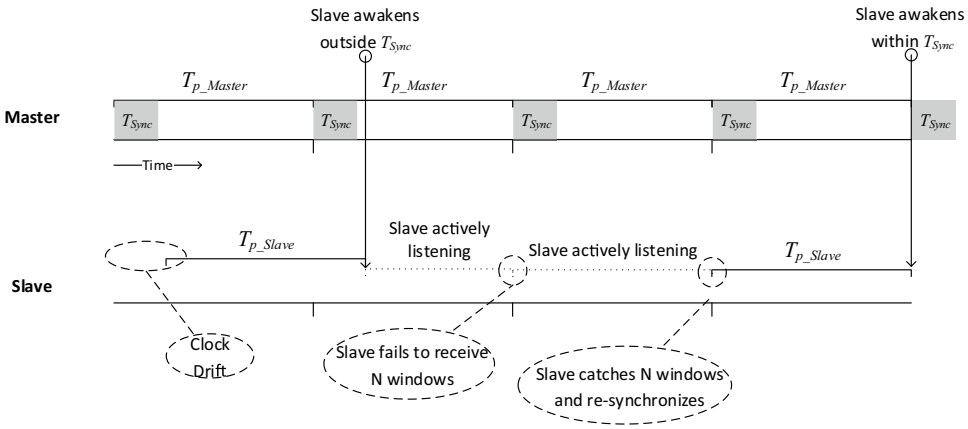

Fig. 4 Worst case scenario where slave misses T_{Sync} and has to actively listen for the sequence. Slave then fails to receive sufficient windows to re-synchronize and actively listens again for another cycle

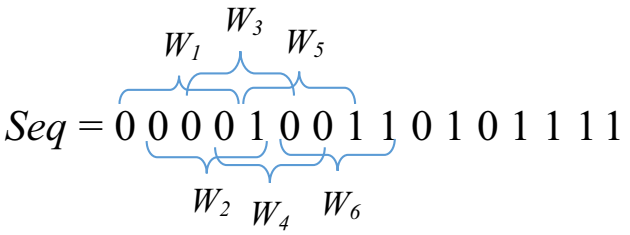

Fig. 5 The De Bruijn sequence Seq has a 16 bit length and a 4-bit sliding window

to 5 *(W_2=0 0 0 1)*; sliding the window one bit at a time the remaining windows are constructed as depicted in Fig. 5. As a reference, Table 1 lists all the possible 4-bit windows from the De Bruijn sequence *Seq*.

Examining this sequence shows that any *4-bit* window has a unique position within the sequence. Hence, in an error-free De Bruijn sequence, it is enough to read 4 bits to determine the current location within the entire 16 bits. An additional feature of a De Bruijn sequence is its ability to easily detect errors. To compensate for the possibility of errors, *m* extra bits are read and error detection is carried out as demonstrated below.

Assuming the transmitted signal is given by *Tx=0 1 0 0 1 1 0*, the De Bruijn windows are W_4, W_5, W_6, and W_7. Due to noise the signal is received as *Rx=0 1 0 0 1 1 1* and decoded as W_4, W_5, W_6, and W_{12} as illustrated in Fig. 6. Since a W_{12} window is received in lieu of a W_7 one, the W_{12} is identified as erroneous.

Given that UART was used for the hardware connection with an *8-bit* transmission unit, an *8-bit* window *256-bit*-long De Bruijn sequence was used as our synchronization sequence. In order to avoid the presence of long sequences of 0 s and 1 s, we decided to use differential Manchester encoding to transmit 01 for a logic 0 value of the De Bruijn sequence and 10 for a logic 1. This encoding scheme has been proven to be effective in several communication fields [33–36]. In contrast, the price of using such a scheme is that it requires double the bandwidth with each *8-bit* De Bruijn window being transmitted as 16 bits.

3.3 Synchronization

For ease of discussion, a parameter naming scheme was adopted to represent the variables in the system. The variables providing time values start with "*T*", while "*L*" is used for bit length amounts. In addition to these, *Seq* is used to represent the De Bruijn sequence in question and W_i represents the window at i^{th} interval in *Seq*. As

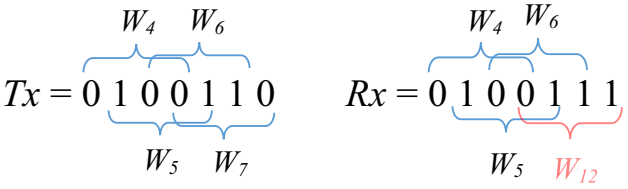

Fig. 6 The *Tx* sequence represents the sent sequence *W4,W5,W6,W7* and *Rx* represents the received sequence with a 1-bit error that breaks the window sequence order with *W12* being received instead of *W7*

Table 1 The list of all the 4-bit windows of the De Bruijun sequence Seq

W_1	0000	W_5	1001	W_9	1010	W_{13}	1111
W_2	0001	W_6	0011	W_{10}	0101	W_{14}	1110
W_3	0010	W_7	0110	W_{11}	1011	W_{15}	1100
W_4	0100	W_8	1101	W_{12}	0111	W_{16}	1000

a recap, the following variable names have already been encountered:

- T_{p_Master}: The periodic time the master waits to send the synchronization sequence.
- T_{p_Slave}: The periodic time at which the slave wakes up to detect the synchronization sequence.
- T_{Sync}: The time required to send the synchronization sequence.

The serial communication scheme dictates the presence of a start and an end bit for every UART packet of data being sent. Thus, the total sequence time is calculated as follows:

- Original sequence length is denoted by L_{Seq} bits.
- Length after Manchester encoding $= 2*L_{Seq}$ bits
- Length of UART packet is symbolized as L_{pkt} bits.
- Number of UART packets being transmitted $= (2*L_{seq})/L_{pkt}$ packets.
- Length with the start and end bits $= (2*L_{seq})/L_{pkt}$ packets of $(1+L_{pkt}+1)$ bits
- Time to send 1 bit is indicated by T_{bit}.
- Time to send 1 packet $= (2+L_{pkt})*T_{bit}$
- Total sequence length, $T_{Sync} = (2*L_{seq})/L_{pkt} * (2+L_{pkt})*T_{bit}$
- Number of UART packets read during state *Receive N Windows,* is denoted by *M*.
- Size of window W_i is represented by L_{wnd}.

Accordingly, every T_{p_Slave} the receiver will wake up and read *M* UART packets ($M*L_{pkt}$ hardware bits), which represent $M*L_{pkt}/2$ De Bruijn bits or *N* sliding windows of L_{wnd} bits each. Recall that each received window has a unique position within the whole De Bruijn sequence. As such, the slave makes sure these windows are ordered in a way consistent with the proposed protocol. It then proceeds to calculate the exact time. The following are used by the receiver to determine the current time.

- The elapsed time since the last synchronization cycle is denoted by Δt.
- $(T_{p\_Slave}—T_{Sync})$ represents the time from the end of the previous cycle (*Seq* has just been sent) till the beginning of the current De Bruijn sequence (new *Seq* just commenced).
- P_i is the position of the last read UART packet in the current sequence with respect to the beginning of the sequence.
- $P_i*(2+L_{pkt})*T_{bit}$ represents the elapsed time since the beginning of the current *Seq*.
- P_{i-1} is the position of the last read UART packet in the previous synchronization sequence.
- $(P_{i-1}*(2+L_{pkt})*T_{bit}—T_{Sync})$ is the time between the synchronization point in the last sequence till the end of the previous T_{p_Slave}. This subtraction calculates the elapsed time since the last synchronization cycle.

Based on the above,

$$\Delta t = \left(T_{p\_Slave} - T_{Sync}\right) + P_i^*\left(2+L_{pkt}\right)^*T_{bit} - \left(P_{i-1}^*\left(2+L_{pkt}\right)^*T_{bit} - T_{Sync}\right)$$

$$\Delta t = T_{p\_Slave} + P_i^*\left(2+L_{pkt}\right)^*T_{bit} - P_{i-1} * \left(2+L_{pkt}\right)^*T_{bit}$$

$$\Delta t = T_{p\_Slave} + \left(P_i - P_{i-1}\right)^*\left(2+L_{pkt}\right)^*T_{bit}$$

Δt is then added to the previous time stamp to calculate the exact time of reception of the latest synchronization packet; $\Delta t = t_i - t_{i-1}$ where t_i is the time of reception of the last packet within the sequence in cycle *i*.

For Illustration purposes, consider $M = 2$ packets and $L_{pkt} = 8$ bits, then the number of Manchester encoded bits of the 2 UART packets $= 2*L_{pkt} = 16\ bits$. Figure 7 provides an illustration of the Manchester encoded sequence *Seq* that is transmitted by the master. The receiver wakes up and receives the first two packets that are highlighted in gray in Fig. 7. Decoding the two received packets recreates the first portion of *Seq* as depicted in Fig. 8.

The number of decoded De Bruijn bits of the 2 decoded packets $= 16/2 = 8\ bits$. Considering a De Bruijn window size of 4 bits, the 8 received decoded bits in Fig. 8 include $N = 5$ windows, namely, W_1, W_2, W_3, W_4, and W_5, refer to Section III.B. Figure 9 provides an illustration of two cycles of synchronization. During the current cycle, the slave receives two packets and t_i marks the time stamp of the reception of the last packet. On the other hand, t_{i-1} marks the time of the last packet received in the previous cycle. Clock correction is carried out as

$$t_i = t_{i-1} + \Delta t, \text{where} \Delta t = T_{p\_Slave} + \left(P_i - P_{i-1}\right)^*\left(2+L_{pkt}\right)^*T_{bit}$$

From Fig. 9, $P_i = 1$, $P_{i-1} = 2$ and Δt is thus $T_{p\_Slave} + (1—2)*(10)*T_{bit} = T_{p\_Slave} - 10*T_{bit}$. As a conclusion, the clock is corrected as $t_i = t_{i-1} + \Delta t = t_{i-1} + T_{p\_Slave} — 10*T_{bit}$. A more comprehensive case study is provided in Section IV that covers all the experimental details of the presented hardware implementation.

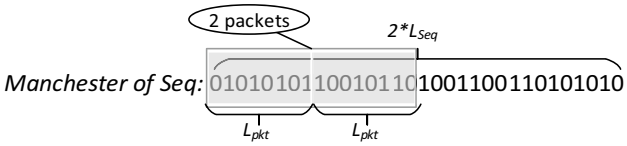

Fig. 7 Manchester encoded *Seq* that is transmitted by the master

Fig. 8 Receiver decodes the received 2 packets to recreate the first portion of *Seq*

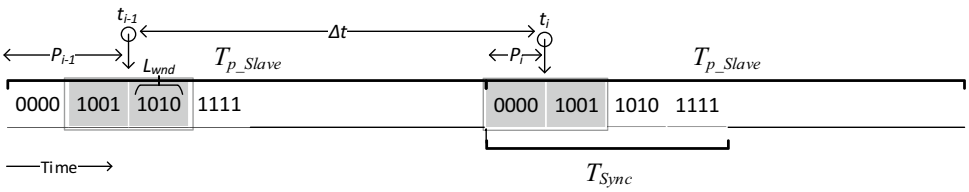

Fig. 9 An illustration of the synchronization mechanism with all the relevant variables

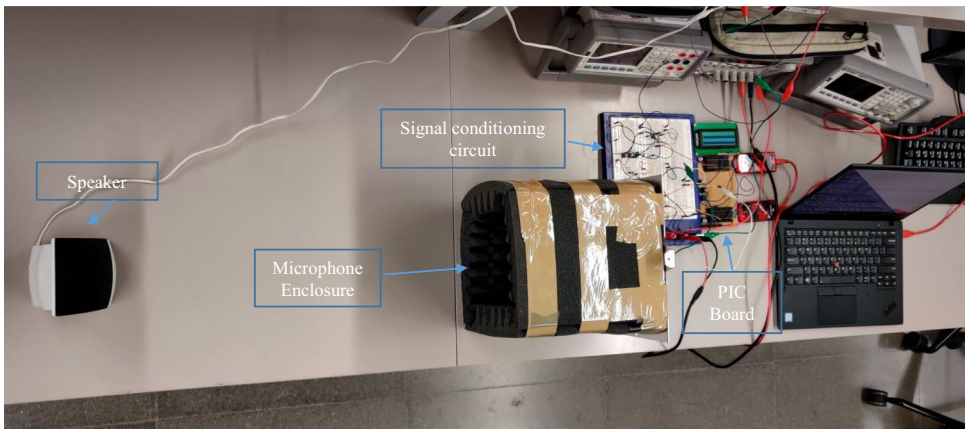

Fig. 10 Top view of the experimental setup showing the processing board of the two processors "PIC Board," the "Signal Conditioning Circuit" and the "Speaker" emitting the synchronization sequence

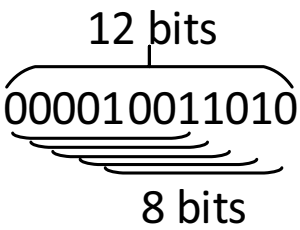

Fig. 11 12 bits generate 5 sliding windows of 8 bits each

4 Experimental evaluation

An experimental case study was developed to showcase the feasibility of the proposed system. The details of the experimental setup and the achieved results are presented below.

4.1 System hardware prototype

To implement the master side of the system, a processing device is connected to an off-the-shelf single channel speaker as depicted in Fig. 10 and implements a sliding window of 8 bits each as shown in Fig. 11. Figure 10 also shows shows the slave side of the proposed system, which includes a microphone, a conditioning circuit, and a processor, as also shown in more details in Fig. 12. The basic speaker-microphone system was tested using a square wave from a function generator and its performance evaluated at different distances and frequencies. As seen in Fig. 12, a very basic electret condenser microphone was used without the need for a high-quality microphone thus lowering the cost of the system. The output of this circuit was then decoupled, amplified, and clipped to extract the original square wave from the received signal and the result is displayed in Fig. 13. It is important to highlight that the upper square wave represents the input to the speaker and that is faithfully recreated at the receiver side as the lower wave form suggests.

Note that the processor used at both the transmitting and receiving sides was the PIC 18F4550 since modest processing power, minimal memory size, and a UART are needed.

4.2 Experimental setup

Figure 10 shows the placement of the speaker and microphone. The experiment was carried out in a noisy lab environment with 10 desktop computers running, a ventilation system, and all the equipment depicted in Fig. 10. In doing this, the setup is closer to a real scenario as opposed

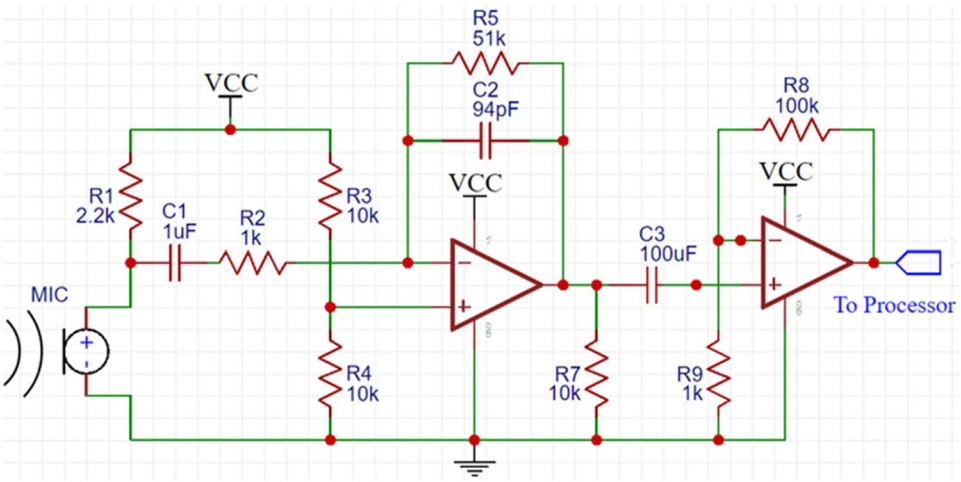

Fig. 12 The signal conditioning circuit

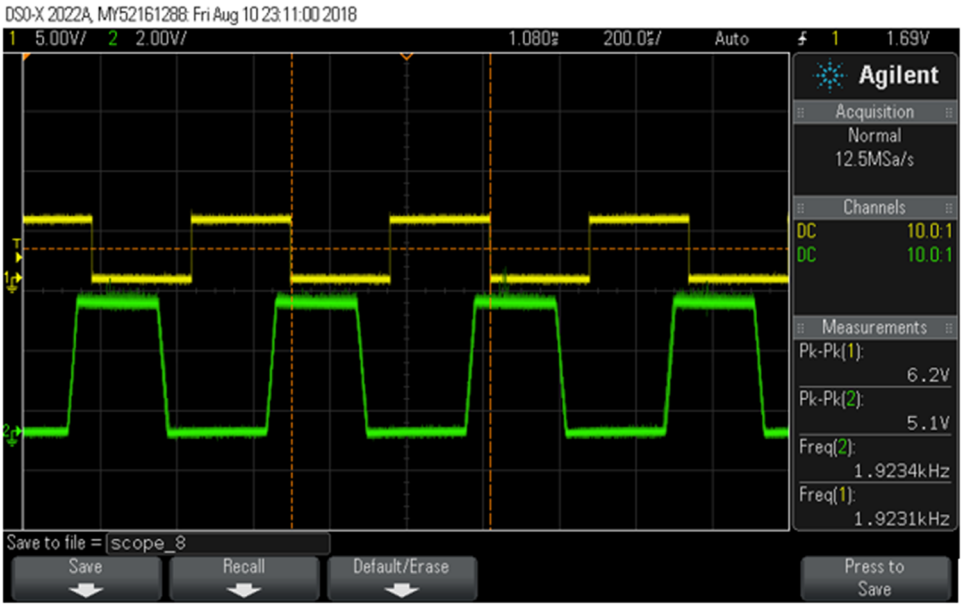

Fig. 13 Receiver performance at bit duration 0.52 ms

to running the experiment in a studio with minimum noise levels.

To study the effect of reflections, five different setups of the microphone were used. The first setup is simply the microphone by itself (referred to as Standalone), the second utilizes acoustic insulation foam behind the microphone (referred to as One Insulation Edge OIE and shown in Fig. 14), the third uses more insulation edges to cover five sides of the microphone (referred to as Five Insulation Edges FIE, Fig. 15), the fourth places a metallic plate behind the microphone (Metal Plate MP, Fig. 16), while the fifth combines the metallic plate with a four-sided boundary of the acoustic insulation (Metal Plate with Four Insulation Edges MPFIE, Fig. 17). Note that the MP and MPFIE were inspired by the boundary effect microphone [37]. To further understand the effect of reflections, the metal plates were replaced with an insulation edge

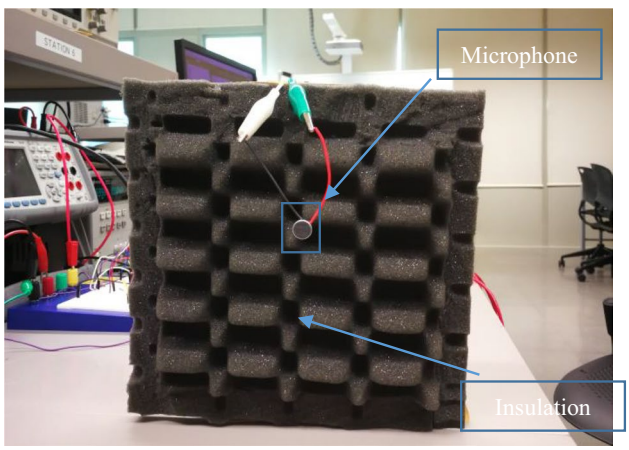

Fig. 14 The microphone in the One Insulation Edge (OIE) configuration

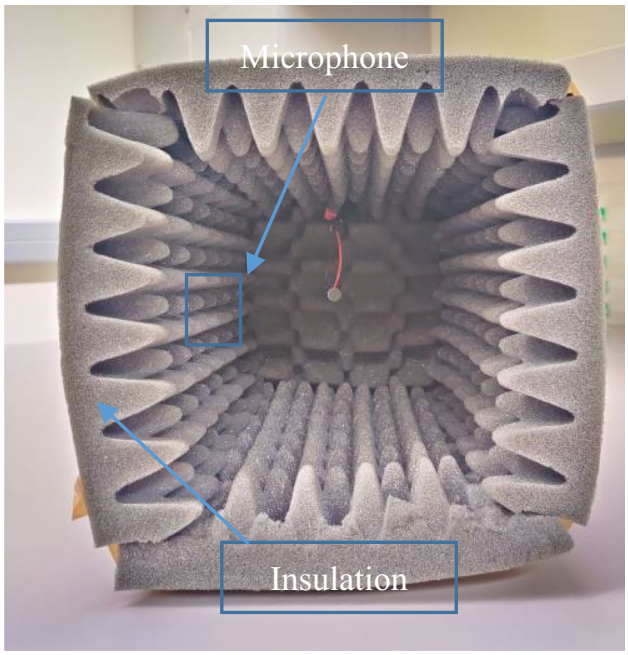

Fig. 15 The microphone in the Five Insulation Edges (FIE) configuration

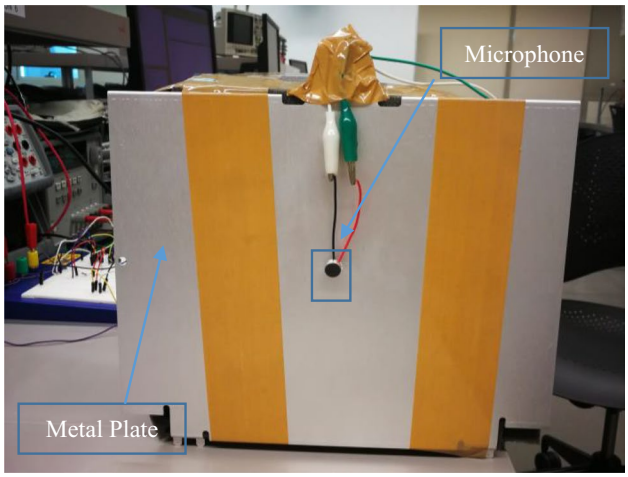

Fig. 16 The microphone in the Metal Plate (MP) configuration

to create the OIE and FIE setups from the MP and the MPFIE setups respectively. The experimental configuration detailed below was used with these five microphone setups and the obtained results are compared and discussed in following section.

By experimentation, the used speaker-microphone configuration exhibited acceptable performance at bit durations ranging from 0.2 to 1 ms, with the best performance being observed at a bit duration of 0.52 ms. Thus, the PIC was programmed to send serial data with a bit duration

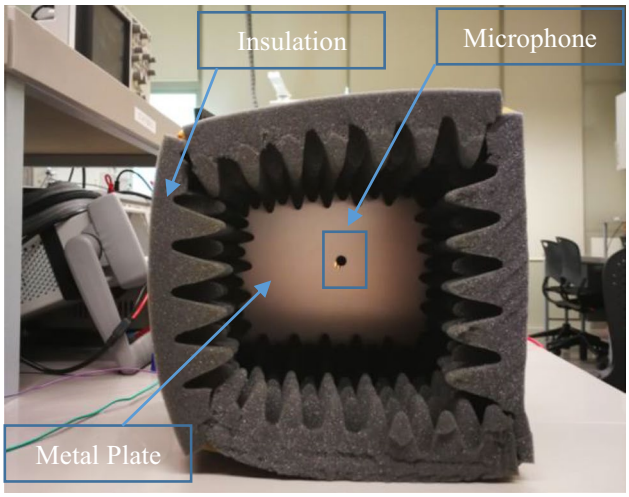

Fig. 17 The microphone in the Metal Plate with Four Insulation Edges (MPFIE)

as close as possible to 0.52 ms as dictated by the utilized crystal and PIC, specifically 0.520833 ms. It is worth noting that the 0-logic level was sent as − 2.5 V and the 1-logic level as 2.5 V to the speaker. The following lists the various configurations of the experiment:

- T_{p_Master}: 60 s
- T_{p_Slave}: 60 s
- Original sequence length, L_{Seq} bits = 256.
- Length after Manchester encoding = 2*L_{Seq} bits = 512 bits.
- Length of UART packet, L_{pkt} bits = 8 bits.
- Number of UART packets being transmitted = (2*L_{seq})/L_{pkt} = 64 packets.
- Length with the start and end bits = (2*L_{seq})/L_{pkt} packets of (1 + L_{pkt} + 1) = 10 bits.
- Time to send 1 bit, T_{bit} = 0.520833 ms.
- Time to send 1 packet = (2 + L_{pkt})*T_{bit} = 5.20833 ms.
- Total sequence length, T_{Sync} = (2*L_{seq})/L_{pkt} * (2 + L_{pkt})*T_{bit} = 333.333 ms.
- Number of UART packets, $M = 3$.
- Size of sliding De Bruijn window, $L_{wnd} = 8$.

Every T_{p_Slave}, the receiver will wake up and read 3 UART bytes (24 hardware bits), representing 12 De Bruijn bits. Since $L_{wnd} = 8$, this maps to $N = 5$ sliding windows as depicted in Fig. 11.

Recall that each received window has a unique position within the whole De Bruijn sequence. As such, the slave makes sure these windows are ordered in a way consistent with the proposed protocol. It then proceeds to calculate the exact time. The following equation is used by the receiver to determine the current time, where Δt is added

to the previous time stamp to calculate the exact time of reception of the latest synchronization packet.

$$t_i = t_{i-1} + \Delta t, \text{where } \Delta t = T_{p\_Slave} + \left(P_i - P_{i-1}\right)^* \left(2 + L_{pkt}\right)^* T_{bit}$$

$$\Delta t = 60000ms + \left(P_i - P_{i-1}\right)^* 5.20833ms$$

The transmission delay is about 2.91 ms per meter of air at 20 °C. Calculating the exact delay depends on the distance between the speaker and microphone. A simple protocol similar to the ones described in [16] and [17] can be utilized at the initialization of the system so that each receiver will know its exact location relative to the speaker and calculate the delay needed for the signal to reach it accordingly.

4.3 Accuracy and performance

To test the performance of the system, a set of 512 packets that carry the De Bruijn sequence is transmitted through the speakers at varying offsets of 10 cm from the receiver circuit. The 512 packets are sent over five trials, one for each microphone setup, and the results are presented in Fig. 18, Table 2, and Fig. 19.

As presented earlier, the receiver should wake-up and attempt to receive 5 sliding widows *(N = 5)* which map to 3 UART packets *(M = 3)*. By checking the bit patterns in the received windows with the De Bruijn sequence, the receiver can deduce if the received windows are correctly following the sequence. The reception of three consecutive correct UART packets provides the needed 5 sliding windows and the receiver has all the information required to re-synchronize. Thus, in the discussion to follow, the reception of 3 consecutive correct UART packets is referred to as the reception of a correct sequence and this also represents a resynchronization.

Figure 18 plots the number of correct sequences that are received from the original 512 packets. The results of five trials representing the different microphone setups are represented by the following labels: 1—Standalone, 2—OIE, 3—FIE, 4—MP, and 5—MPFIE. The setups are depicted in (Fig. 14 to Fig. 17).

The plotted results in Fig. 18 prove that the Standalone and FIE setups are the worst performers as they provided the least amount of correct sequences at all distances. In the results for distances that are at 80 cm and below, the OIE setup provided the largest amount of received correct sequences, while the metal plate helped in providing the better results for distances of 90 cm and above. The best performer in the 90 cm and above separation is the MPFIE setup. It can be concluded that at the larger distances, the reflections from the metal plate helped in reconstructing the signal in a similar fashion to a boundary effect microphone, while at smaller distances, the insulation helped in minimizing the effect of the destructive reflections.

A reception of a correct sequence implies resynchronization. Thus, from the above presented experimental values, it is obvious that there are resynchronization opportunities at all the separation distances up to 250 cm which is the limit of reliable reception of correct sequences. The issue to tackle now is the mean time required for a receiver to wake up and receive a correct sequence to be able to synchronize. Especially when using the OIE setup for distances 80 cm and below and the metal plate setups (MP or MPFIE) for distances larger than that.

When a receiver wakes up to synchronize, it faces two situations. Either there is a sequence being transmitted or not. In case there are no sequences being transmitted, the receiver needs to stay awake for a maximum of T_{p_Master} till the next synchronization sequence arrives. If the receiver awakes and the sequence is being transmitted, then some packets have already been missed. The receiver has to wait till it receives 3 correct UART packets (a correct sequence) to be able to re-synchronize. For the five microphone setups, 512 UART packets have been sent and the received packets were recorded on the receiver's side. These recorded packets

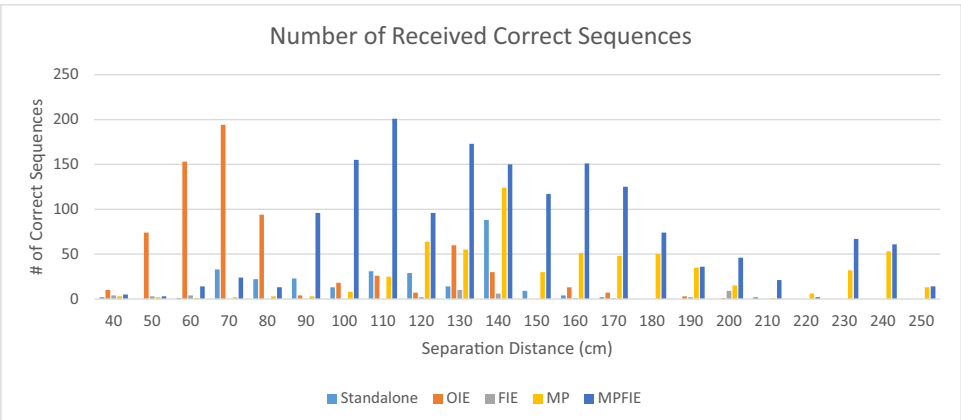

Fig. 18 The number of received correct sequences from the set of 512 sent packets (510 total sequences). A correct sequence is three consecutive correct received UART packets. The three packets are needed to derive the location of the bit pattern in the De Bruijn sequence and thus lead to resynchronization. Any data point on the plot above signifies a successful synchronization instance

Table 2 The mean time till successful re-synchronization at the different distances and for the five microphone setups. The "No Sync." entry denotes the cases in which re-synchronization was not successful. For distances below 90 cm, the OIE setup was always successful while the MP and MPFIE setups were always successful for distances 90 cm and above

Separation Distance (cm)	Mean Time Till Successful Synchronization (ms)				
	Standalone	OIE	FIE	MP	MPFIE
40	841.16157	536.18859	464.39976	15.62499	502.74019
50	No Sync.	109.09491	892.01641	26.78570	15.62499
60	31.24998	43.73767	521.18992	15.62499	331.83469
70	115.36506	29.35325	No Sync.	15.62499	376.66463
80	132.87482	64.94008	No Sync.	15.62499	475.58030
90	252.76060	461.93275	No Sync.	15.62499	92.51786
100	244.45049	125.94689	No Sync.	21.63460	52.36023
110	153.23238	161.06166	No Sync.	337.94653	44.41434
120	214.78580	432.42756	367.22501	73.10873	65.32968
130	299.23501	71.03014	263.60504	79.53981	39.79041
140	64.35181	94.34329	276.75480	46.61901	45.12853
150	430.28058	No Sync.	No Sync.	86.66082	79.36552
160	416.86189	149.30159	39.06248	96.93634	35.03786
170	908.47816	361.68445	41.66664	111.04322	44.51687
180	No Sync.	No Sync.	No Sync.	90.24354	73.63826
190	No Sync.	401.16008	455.06013	121.32147	291.06602
200	No Sync.	677.08290	274.90975	255.83789	114.45999
210	589.11037	No Sync.	No Sync.	15.62499	176.77555
220	No Sync.	No Sync.	No Sync.	589.00450	885.43155
230	No Sync.	No Sync.	No Sync.	145.25705	88.71659
240	No Sync.	No Sync.	No Sync.	101.97072	86.84810
250	No Sync.	No Sync.	No Sync.	244.45582	223.77742
				Scheme successful at all distances	Scheme successful at all distances

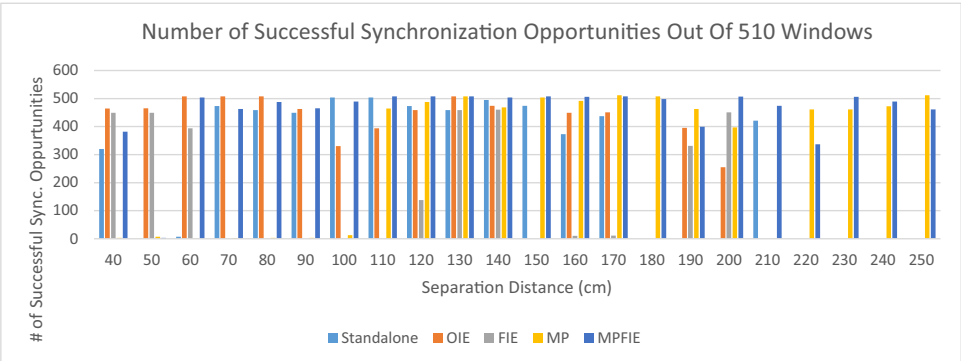

Fig. 19 If the receiver wakes up while the sequence is being transmitted and still able to receive a correct sequence, then that is counted as a successful synchronization. There are 512 packets being sent which amount to 510 opportunities to synchronize, the number of successful opportunities is presented in the chart

provide a timeline of what the receiver would see when it wakes up. Each packet reception initiated a counter as to when a correct sequence would be received. That timing data was gathered for all the reception slots that would receive a correct sequence and the mean time to synchronize is presented in Table 2. It is to be noted that the slots of reception that occur after the last correct sequence is received will have to wait for T_{p_Master} before being able to synchronize. The data points denoted by "No Sync." in Table 2 highlight some cases that would not be able to synchronize. There are multiple cases for the Standalone, OIE, and FIE setups, one case for the MP setup, and no such case for the MPFIE setup. If we assume that a metal plate will be used (MP and MPFIE) for distances 90 cm and above, and that an

insulating plane will be used (OIE) for distances lower than 90 cm, then the OIE, MP, and MPFIE setups will always synchronize. Synchronization will occur in either the current synchronization sequence or the one after. Figure 19 plots the number of successful synchronizations within the 510 windows when the receiver wakes up while the sequence is being transmitted. A receiver that wakes up at exactly the time the sequence is being received has the chance to benefit from all the presented opportunities. A receiver that wakes in the middle of the transmission has a higher chance of missing the successful receptions. In all aspects, the receiver will be able to synchronize in either the current cycle or the one next (the receiver will be actively awake when waiting for the next cycle and thus can benefit from all the opportunities). Given the right choice of the microphone setup, the receiver will always synchronize up to a distance of 250 cm.

Fig. 20 A cardboard obstacle placed in the middle of the path between the speaker and microphone

There are multiple aspects that can be utilized in the system to further enhance these results and provide implementations that can successfully increase the distance of separation significantly beyond the 250 cm. These aspects would include better components on the microphone and speaker side: particularly speakers that can handle the abrupt nature of a square wave. Another important aspect is the introduction of bit error correction to the scheme. Error correction can benefit from redundant data and/or the predictable nature of the De Bruijn sequence data.

4.4 Indirect, non-line of sight, and noisy environment results

The results provided earlier focused on highlighting the ability to synchronize with increasing separation distance between the speaker and microphone under varying setups while keeping a direct line of sight and having the speaker and microphone squarely facing each other. This section provides data of three different experiments, the first has the planes of the microphone and speaker at a 45 and 90 degrees instead of being parallel, the second provides data with an obstacle breaking the line of sight as shown in Fig. 20, and the third runs the experiment while noisy generator units are running.

4.4.1 Indirect experiment

The results of the experiment are presented in Table 3. It can be seen that the number of correct sequences received is diminished when compared with the original case presented in the previous section. This decrease does not prevent

Table 3 The results of the experiment with the planes of the microphone and speaker at an angle of 45 degrees and 90 degrees respectively. The results prove that synchronization is still possible but with lower performance especially with the 90 degrees case

		# correct sequences					Mean Time Till Successful Synchronization (ms)					# successful syncs out of 510 slots				
Angle (deg)	Distance (cm)	MPFIE	FIE	Stand alone	MP	OIE	MPFIE	FIE	Standalone	MP	OIE	MPFIE	FIE	Stand alone	MP	OIE
45	100	123	2	14	4	0	49.08	15.62	325.40	15.62	No Sync.	512	2	378	4	0
	150	59	4	0	20	7	74.56	467.58	No Sync.	407.17	413.06	489	512	0	333	273
	200	22	0	15	19	0	243.59	No Sync.	242.41	250.37	No Sync.	400	0	458	448	0
	250	20	0	0	51	0	138.53	No Sync.	No Sync.	93.66	No Sync.	437	0	0	488	0
	300	6	0	0	7	0	277.79	No Sync.	No Sync.	401.57	No Sync.	337	0	0	373	0
90	100	0	0	3	23	0	No Sync.	No Sync.	689.08	176.92	No Sync.	0	0	498	474	0
	150	4	0	0	48	1	293.24	No Sync.	No Sync.	187.76	18.23	331	0	0	467	2
	200	0	0	0	5	1	No Sync.	No Sync.	No Sync.	449.25	39.06	0	0	0	363	10
	250	0	0	0	16	0	No Sync.	No Sync.	No Sync.	231.85	No Sync.	0	0	0	322	0
	300	0	0	0	12	0	No Sync.	No Sync.	No Sync.	360.36	No Sync.	0	0	0	474	0
					Scheme successful at all distances					Scheme successful at all distances					Scheme successful at all distances	

synchronization especially with a metal plate. With both the MP and MPFIE setups a significant number of correct sequences is received and synchronization is successful for all the considered distances with a 45 degrees angle (except for the 50-cm case for MP). With the 90 degrees case, only the MP setup was successful with synchronization at all the distances. Table 3 also presents the mean time till successful synchronization and the number of successful synchronization out of the 510 slots.

4.4.2 Obstacle

To verify the operation of this synchronization mechanism where a no direct line of sight is available, an obstacle was placed between the speaker and microphone as depicted in Fig. 20. The obstacle did diminish the number of correct sequences received as reported in Table 4, but the circuit was still successful in synchronizing between 150 and 300 cm separation in the MP setup. As can be deduced from the results, the insulation had a negative effect on the overall performance in such a case.

4.4.3 Noisy environments

To study the effect of noisy environments on the performance of the system, we conducted the experiment in an electric power systems lab area that has significantly noisy generator units using the MP configuration. Table 5 provides the number of received correct sequences with the generator units on and off. The provided results clearly show the detrimental effect of noise on the number of received sequences as there is a large observed decrease in the number of sequences in the noisy case.

Table 5 The number of received correct sequences from the set of 512 sent packets (510 total sequences). A correct sequence is three consecutive correct received UART packets. The three packets are needed to derive the location of the bit pattern in the De Bruijn sequence and thus lead to resynchronization. The noise from the generator units significantly decreased the number of these received correct sequences

Separation distance (cm)	Number of received correct sequences (MP)	
	Generators off	Generators on
100	2	0
110	6	0
120	6	0
130	7	5
140	7	4
150	11	10
160	24	4
170	21	2
180	24	8
190	20	8
200	18	11
210	27	2
220	37	8
230	50	24
240	54	19
250	53	18

5 Concluding remarks

This work attempts to create a wireless synchronization scheme for simple extant smart devices using sound. The proposed scheme synchronizes multiple devices in one room from one master in a broadcast fashion. One of the advantages of this scheme is that synchronization is carried out in an offline fashion with only the separation distance as a restriction on the physical positioning of the devices. Moreover, the hardware requirement is a minimum. On one hand, the master only needs access to a speaker and thus any PC can initiate synchronization. On the other hand, the deployed devices only need to have a UART, a microphone, and a conditioning circuit (refer to Fig. 12).

Another aspect of this design is using serial communication instead of sampling with analog to digital converters. The received signal is amplified and clipped using basic

Table 4 The results of the experiment with an obstacle in the path between the speaker and microphone that prevents a direct line of sight. The results prove that synchronization is still possible but with lower performance.

		# correct sequences					Mean Time Till Successful Synchronization (ms)					# successful syncs out of 510 slots				
Obstacle	Distance (cm)	MPFIE	FIE	Stand alone	MP	OIE	MPFIE	FIE	Standalone	MP	OIE	MPFIE	FIE	Stand alone	MP	OIE
	150	0	0	0	12	0	No Sync.	No Sync.	No Sync.	207.27	No Sync.	0	0	0	347	0
	200	2	0	0	12	1	445.63	No Sync.	No Sync.	432.38	1346.35	319	0	0	512	512
	250	3	0	0	6	0	432.21	No Sync.	No Sync.	702.87	No Sync.	307	0	0	365	0
	300	0	0	1	17	0	No Sync.	No Sync.	54.69	441.42	No Sync.	0	0	16	476	0
					Scheme successful at all distances					Scheme successful at all distances					Scheme successful at all distances	

circuit design and read as digital data by the processor as seen in Fig. 12. Dropping the analog to digital sampling of the signal allowed for relaxing the memory requirements. In fact, the authors of [13] specified that in order to handle the memory requirements of heavy sampling, their approach had to rely on an online solution. Another advantage of dropping the analog to digital sampling is that some processors do not have built-in analog-to-digital converters and rely on external samplers which might not be available in many devices. However, most processors do have a UART or SCI port. Finally, using a serial port is less computationally complex than data sampling and the accompanying calculations.

Using a basic processor and a regular speaker also means that a transmitter can be created on a personal computer without the need for an independent device. The transmitter code can be created as a simple script that can be easily run on the computer. Once this script is activated, the data will be transmitted using the preexisting computer speakers.

Overall, the presented system was able to reliably transmit and receive the synchronization sequence over a distance of 250 cm. If we consider the MPFIE setup, then the mean synchronization time is 183.14 ms for the case when the receiver wakes up during a synchronization sequence for all separation distances or has an upper bound of approximately T_{p_Master} for the case when the receiver wakes and is required to wait till the next synchronization sequence. The system, especially with the MP setup, was successful in synchronization when the planes of the microphone and speaker were not parallel and also with an obstacle that blocks the line of sight.

6 Future work

Two main areas are to be considered in enhancing the work presented in this manuscript; the first is extending the range of the system, and the second is working in insecure areas to prevent among others a false sequence from being presented to the system. The utilized constraints of cheap and simple hardware such as the electret microphone with the conditioning circuit and utilizing a UART in handling the physical layer communication had an adverse effect on the range of the system. Relaxing these constraints by using more specialized speakers, microphones, and sound cards along with utilizing bit error correction or modulation techniques could help in increasing the distance over which reliable synchronization sequence communication can be attained. To test this assumption, we used Frequency Shift Keying to encode a "0" as a 18 kHz sinusoid and a "1" as a 20 kHz sinusoid in a similar fashion to what was presented in [38] which also helped us in working with inaudible signals. We also relaxed all the hardware constraints and utilized the microphone and soundcard of a Thinkpad Carbon X1 to receive the signal and demodulate it with a Matlab code. The bit duration was extended dramatically from 0.52 to 100 ms. This new setup helped in reliably receiving synchronization sequences in ideal (line-of-sight, no obstacles, and no major noise sources) outdoor and indoor environments with a 20-m and 30-m separation respectively. We will build on this outcome to gradually increase the hardware complexity from what was presented in this manuscript and work towards achieving the ideal balance between cost, speed, and range in more demanding environments.

The other area for improvement is security. The method presented in the manuscript broadcasts the synchronization sequence openly where it is deployed. In case a potential attacker gains physical access to the area, they can instigate a man in the middle attack. Encryption of the sequence between sender and receiver should help in mitigating some of the security concerns of openly broadcasting the synchronization sequence. Other techniques should also be studied to prevent an attacker from simply recording the sequence and transmitting it at varying intervals.

Acknowledgements The authors acknowledge the efforts of Aymane El Baarini, Christelle Saliba, Rayan Al Sobbahi in conducting portions of the lab experiments.

Funding This project has been jointly funded with the support of the National Council for Scientific Research in Lebanon CNRS-L and the Lebanese American University.

References

1. Kim JY, Lee HY, Son JY and Park JH (2015) "Smart home web of objects-based IoT management model and methods for home data mining," in 17th Asia-Pasific Network Operations and Management Sumposium (APSNOMS)
2. Li F, Yang Y, Chi Z, Zhao L, Yang Y, Luo J (2018) "Trinity: Enabling self-sustaining WSNs indoors with energy-free sensing and networking." ACM Trans Embedded Comput Syst 17(2):Article 57
3. Guo H, Crossly P (2017) Design of a time synchronization system based on GPS and IEEE 1588 for trasnmission substations. IEEE Trans Power Delivery 32(4):2091–2100
4. IEEE Guide for Designing a Time Synchronization System for Power Substations, IEEE Std 2030.101TM-2018
5. IEEE Standard Profile for Use of IEEE 1588TM Precision Time Protocol in Power System Applications, IEEE Std C37.238TM-2011
6. Amelot J and Stenbakken G (2012) "Testing phasor measurement units using ieee 1488 precision time protocol," in Conference on Precision Electromagnetic Measurements (CPEM) 2012
7. Mills D (1991) Internet time synchronization: the network time protocol. IEEE Trans Commun 39(10):1482–1493
8. Helling D, Hense M, Van der Auweraer H and Leuridan J (2005) "Data stream synchronization of distributed measurements systems using GPS technology," in IEEE Intelligent Data Acquisition and Advanced Computer Systems: Technology and Applications IDAACS 2005

9. Refan MH and Valizadeh H (2011) "Redundant GPS time synchronization boards for computer networks," in 19th Telecommunications Forum (TELFOR) Proceedings of Papers 2011
10. Yan L (2012) "Application of GPS technology in frame-signal synchronization system of wireless broadband access," in 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet) 2012
11. Li L, Xing G, Sun L, Huangfu W, Zhou R and Zhu H (2011) "Exploiting FM radio data system for adaptive clock calibration in sensor networks," in Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services
12. IEEE Std802.15.4TM-2015, IEEE Standart for Low-Rate Wireless Personal Area Networks (WPANs)
13. Guo X, Mohammad M, Saha S, Chan CM, Gilbert S and Leong D (2016) "PSync: visible light-based time synchronization for Internet of Things (IoT)," in IEEE INFOCOM The 35st Annual IEEE International Conference on Computer Communications
14. Yang Q, An D and Yu W (2013) "On time desynchronization attack against ieee 1558 protocol in power grid systems," in IEEE Energytech 2013
15. Shijith N, Poornachandran P, Sujadevi VG and Dharmana MM (2017) "Spoofing technique to counterfeit the gps receiver on a drone," in International Conference on Technological Advancements in Power and Energy
16. Bonebrake C, O'Neil LR (2014) Attacks on GPS time reliability. IEEE Secur Priv 12(3):82–84
17. Psiaki ML, Humphreys TE (2016) GNSS spoofing and detection. Proc IEEE 104(6):1258–1270
18. Humphreys TE, Ledvina BM, Psiaki ML, O'Hanlon BW and Kintner Jr PM (2008) "Assessing the spoofing threat: development of a portable GPS civilian spoofer," in 21st International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2008)
19. Aoki N, Ikeda K and Yasuda H (2020) "A synchronization technique using De Bruijn sequences for inaudible sound communication systems," in International Conference on Emerging Technologies for Communications (ICETC 2020), online, December 2–3
20. "Send data using sound," Chirp, [Online]. Available: https://www.chirp.io
21. Nguyen Q, Choi J (2016) Matching pursuit based robust acoustic event classification for surveillance systems. Comput Electr Eng 57:43–54
22. Qiao G, Bilal M, Liu S, Babar Z, Ma T (2018) Biologically inspired covert underwater acoustic communication - a review. Physical Communication 30:107–114
23. Azad S, KhandakerTabin H, Nandi D, Pathan A-SK (2015) A high-throughput routing metric for multi-hop underwater acoustic networks. Comput Electrical Eng 44:24–33
24. Morns IP, Hinton OR, Adams AE and Sharif BS (2001) "Protocol for sub-sea communication networks," in Proceedings of MTS/IEEE Oceans Conference 2001. An Ocean Odyssey
25. Hong F, Yang B, Zhang Y, Xu M, Feng Y and Guo Z (2014) "Time synchronization for underwater sensor networks based on multi-source beacon fusion," in 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)
26. De Bruijn N (1946) A combinatorial problem. Proceedings of Nederlandse Akademie van Wetenschappen 49:758–764
27. Mitchell CJ, Etzion T, Paterson KG (1996) A method for constructing decodable de Bruijn sequences. IEEE Trans Inform Theory 42(5):1472–1478
28. Margossian H, Sayed MA, Fawaz W, Nakad Z (2019) Partial grid false data injection attacks against state estimation. Int J Electr Power Energy Syst 110:623–629
29. Spinsante S, Andrenacci S and Gambi E (2011) "Binary De Bruijn sequences for DS-CDMA systems: analysis and results". EURASIP J Wireless Commun Netw 4
30. Sacchi C (2019) "About the use of a new set of quadriphase sequences for increasing security of PMR over LTE primary synchronization," in IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Sochi
31. Chee YM, Etzion T, Kiah HM, Khu Vu V and Yaakobi E (2019) "Constrained de Bruijn codes and their applications," in IEEE International Symposium on Information Theory (ISIT), Paris
32. Howie RM, Paxman J, Bland PA, Towner MC, Sansom EK, Devillepoix HAR (2017) Submillisecond fireball timing using de Bruijn timecodes. Meteorit Planet Sci 52(8):1669–1682
33. Forster R (2000) "Manchester encoding: opposing definitions resolved". Eng Sci Educ J 278–280
34. Jose J (2013) "Design of Manchester II bi-phase encoder for MIL-STD-1553 protocol," in International Multi-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMacs4) 2013
35. "Manchester Encoding Basics," [Online]. Available: http://ww1.microchip.com/downloads/en/AppNotes/Atmel-9164-Manchester-Coding-Basics_Application-Note.pdf. Accessed 17 Aug 2021
36. Tao Q, Zhong C, Lin H and Zhang Z (2018) "Symbol detection of ambient backscatter systems with manchester coding". IEEE Trans Wireless Commun 17(6)
37. Capel V (2016) Newness audio and Hi-Fi, engineer's pocket book. Elsevier
38. Guri M, Solewicz Y and Elovici Y (2018) "MOSQUITO: covert ultrasonic transmission between two air-gapped computers using speaker-to-speaker communication," in Proceedings of the 2018 IEEE Conference on Dependable and Secure Computing (DSC), Kaohsiung, Taiwan (ROC)