Wireless Sensor Network Synchronization

Feasibility study for application within a network ofMuse platforms

Revision history

Date	Revision	Author	Description
2016/06/06	1	Daniele Comotti	First draft

1 Introduction

This document is a brief summary of major techniques used to synchronize nodes within the same wireless sensor network, which could be suitable for applications of Muse platforms where alignment in time of data is a key requirement.

2 Problem description

The Muse platform can be used within a wireless network based on a "star" topology with either a computer or a mobile phone acting as central node. In this application, data coming out from each device must be carefully aligned in time with the proper accuracy if any massive post-processing is foreseen.

The clock of each electronic device is derived from an oscillator. In this case, the Muse mounts an oscillator provided by Murata with the following specs [1]:

- 8 MHz oscillation frequency;
- 0.5 % frequency tolerance;
- $\pm$0.2% frequency shift by temperature between -40 °C (-40 F) and 80 °C (176 F).;
- $\pm$0.1% frequency aging.

Therefore, each node (central node included) of the network features its own local clock, which can be described as follows [2]:

$$C_i(t) = a_i \cdot t + b_i$$

where *i* denotes the i-th node in the network, *t* is time, a_i is the clock drift and b_i is the clock offset.

As a matter of fact, it would be nice to align data of each Muse platform with respect to the central node (node 0):

$$C_i(t) = a_{i,0} \cdot t + b_{i,0}$$

where $a_{i,0}$ is the relative drift and $b_{i,0}$ is the relative offset.

This job is solved by exchanging timestamps. However, each timestamp sent through the network experiments a variable amount of delay which prevents the receiver from knowing exactly when it was transmitted and therefore synchronizing the clock with the proper accuracy. The sources of error are typically the following:

- Send time: time spent to build a message (OS overhead and transfer time to the network interface).
- Access time: delay before the actual transmission (typically occurring at the MAC layer).
- Propagation time: time spent in propagation of the message between the network interface and the sender.
- Receive time: time needed for the network interface of the receiver to receive the message and transfer it to the host.

Moreover, in this particular scenario, the following points must be taken into account:

- Nodes in the network are connected to the central node by means of a SPP connection via the Bluetooth V3 link. The serial port virtualization leads to an intrinsic delay due to data transmission given by its baud-rate, which is 115200 b/s (parity bit included). This means that sending 10 bytes takes at least 600 μs.
- Timestamp on-board of each platform can be easily achieved with a resolution of 1 ms by means of the so called "Systick" interrupt. Increase the resolution is feasible, but maybe not necessary. The date and time provided by the Real Time Clock (RTC) integrated on-board can be used to set a coarse offset to each timestamp which is then carefully adjusted during syncing.
- The central node must provide a timestamp with a resolution at least equal to the other nodes'.

Some remarks:

- Accuracy: describes how close a measurement is to its real value.
- Precision: describes the repeatability of the measurements.
- Resolution: describes how many digits are used in the measurements (having an accuracy of 1 second with a resolution of 1 ms may be not necessary).

3 Synchronization Method

To better understand the problem of time synchronization, the following methods have been investigated:

- Network Time Protocol (NTP), one of the oldest protocols currently used in the Internet [3].
- Reference Broadcast Synchronization (RBS) [2], based on the idea of a "third party" node used to synchronize the others.
- Timing-Sync Protocol for Sensors Networks (TPSN) [2], based on a hierarchical method where each level synchronizes with the higher one.
- Tiny-Sync and Mini-Sync [4], a method which relies on linear programming techniques to solve the synchronization problem.
- Lightweight Time Synchronization (LTS) [5], based on a pair-wise synchronization method.

Starting from the investigated methods, a synchronization technique based on the sender-receiver approach is proposed in the following.

3.1 Clock drift estimation

By referring to the diagram shown in Figure 1, the clock drift can be derived by sending out at time T1 a very short message from the central node to the i-th node.

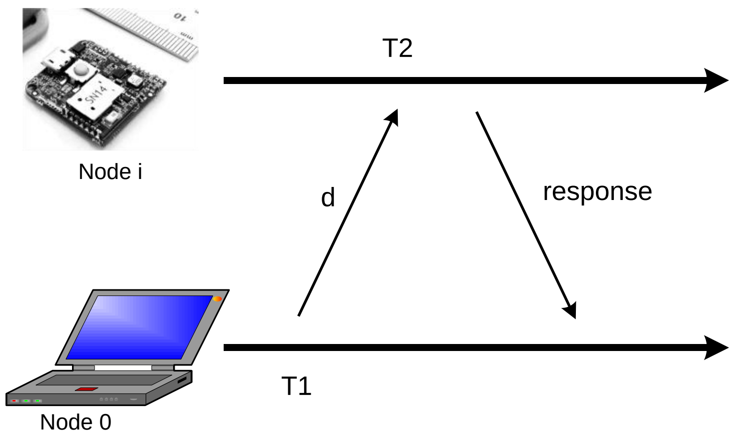

FIGURE 1: TWO WAYS MESSAGE EXCHANGE BETWEEN THE CENTRAL NODE AND A MUSE PLATFORM.

The message takes an amount of time equal to "d" to be transmitted. Therefore, T2 can be rewritten as follows:

$$T_2 = T_1 \cdot a_{i,0} + b_{i,0} + d$$

Once the i-th node receives the message, it replies with a response message carrying the timestamp T2. Once the response is received, the central node knows T2.

Later on, the same procedure is done and another pair of timestamps (see Figure 2), T3 for the central node and T4 for the i-th node, is collected:

$$T_4 = T_3 \cdot a_{i,0} + b_{i,0} + d$$

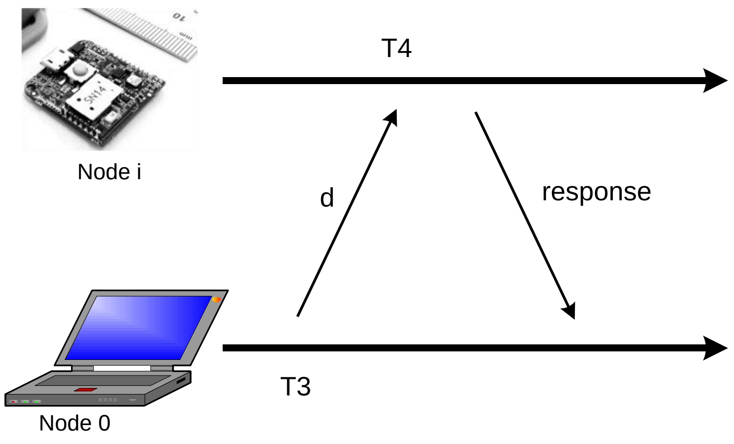

FIGURE 2: YET ANOTHER TWO WAYS MESSAGE EXCHANGE.

The clock drift of the i-th node can be computed as follows:

$$\frac{T_4 - T_2}{T_3 - T_1} = \frac{T_3 \cdot a_{i,0} + b_{i,0} + d - T_1 \cdot a_{i,0} - b_{i,0} - d}{T_3 - T_1} = \frac{T_3 \cdot a_{i,0} - T_1 \cdot a_{i,0}}{T_3 - T_1} = \frac{a_{i,0}(T_3 - T_1)}{T_3 - T_1} = a_{i,0}$$

3.2 Clock offset estimation

Once the clock drift of the i-th node is known, the offset can be estimated through the overall message exchange ofFigure 3, by taking into account all the 4 timestamp values. It has to be noticed that during this phase the clock drift on the i-th node is updated with the value computed during the previous step.

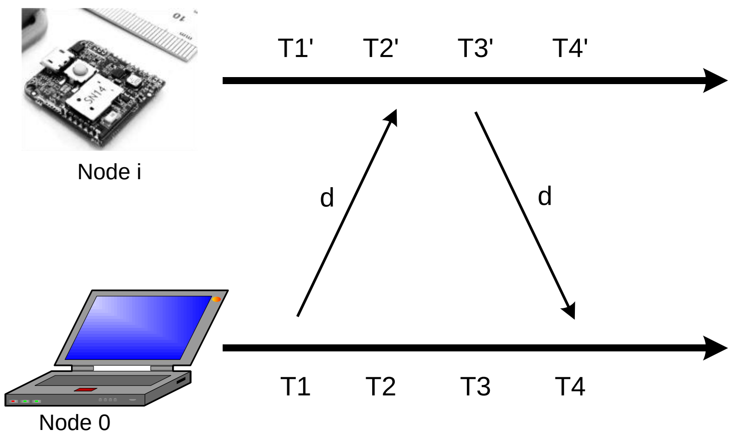

FIGURE 3: TWO WAYS MESSAGE EXCHANGE AFTER ADJUSTING THE CLOCK DRIFT ON THE I-TH CLOCK.

The offset can be computed as follows:

$$\frac{(T_3' + T_2') - (T_4 + T_1)}{2} = \frac{\left(T_3 + b_{i,0} + T_1 + b_{i,0} + d\right) - (T_3 + d + T_1)}{2} = \frac{T_3 + b_{i,0} + T_1 + b_{i,0} + d - T_3 - d - T_1}{2} = \frac{b_{i,0} + b_{i,0}}{2} = b_{i,0}$$

4 Measurement results

4.1 Setup

The test setup is based on the following blocks:

- Laptop computer with Windows 8.1 OS and integrated BT module.
- One Muse platform.
- Customized Muse firmware with an additional protocol of 2 bytes commands and interrupt triggered for a faster communication than usual.
- Computer side software sending out the commands for time sync.

The following static class has been implemented on computer side to achieve a higher resolution clock reference with respect to the usual DateTime class.

```
publicstaticbool IsAvailable { get; privateset; }

    [DllImport("Kernel32.dll", CallingConvention = CallingConvention.Winapi)]
privatestaticexternvoid GetSystemTimePreciseAsFileTime(outlong filetime);

publicstaticDateTime UtcNow
    {
get
        {
if (!IsAvailable)
            {
thrownewInvalidOperationException(
"High resolution clock isn't available.");
            }

long filetime;
            GetSystemTimePreciseAsFileTime(out filetime);

returnDateTime.FromFileTimeUtc(filetime);
        }
    }

static HighResolutionDateTime()
    {
try
        {
long filetime;
            GetSystemTimePreciseAsFileTime(out filetime);
            IsAvailable = true;
        }
catch (EntryPointNotFoundException)
        {
// Not running Windows 8 or higher.
IsAvailable = false;
        }
    }
```

4.2 Clock drift

The first set of measurements has been taken with a relatively small amount of time (50 ms) between T1 and T3 (refer to Figure 1 and Figure 2). However, in such a configuration a lot of variability in the estimated drift has been observed. As shown in Figure 4, not only the dispersion of data in one single measurement is high, but also the dispersion between one measurement and the other.The estimated value of the clock drift is however close to the real one (not yet known for certain), but not sufficiently.

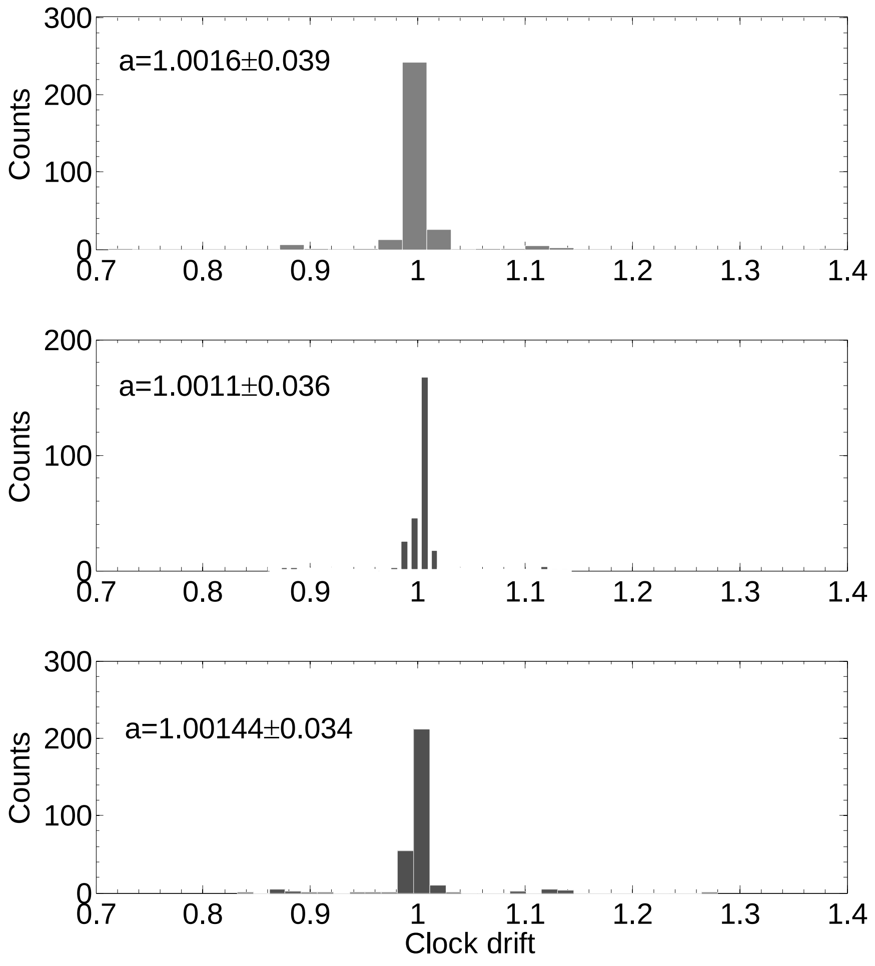

FIGURE 4:HISTOGRAM OF 3 DIFFERENT MEASUREMENTS BASED ON 300 SAMPLES TAKEN WITH 50 MILLISECONDS ONE FROM EACH OTHER.

The variability in the data is due to a quite large and variable amount of time needed to access the Bluetooth link, assessed in tens of milliseconds (10 -30 ms) between T1 and T2 (hereafter called "sampling time"). Therefore, a study has been carried out by increasing the sampling time from 50 ms up to 10 secs, in order to prove such a behavior and find a trade-off value between time taken to carry out the estimation and the related accuracy (the larger the sampling time, the better the accuracy, for any kind of unknown latency has a lower impact).

Results are summarized in Figure 5. It can be noticed that, as the sampling time (T2-T1) increases, the clock drift average approaches the actual value and the standard deviation of the measurement (referred to as error) drops to almost zero. In particular, for 10 seconds of difference, an error of 0.009% has been achieved (corresponding to a sync error of about 0.3 ms after 1 hour, 2 seconds after one year).

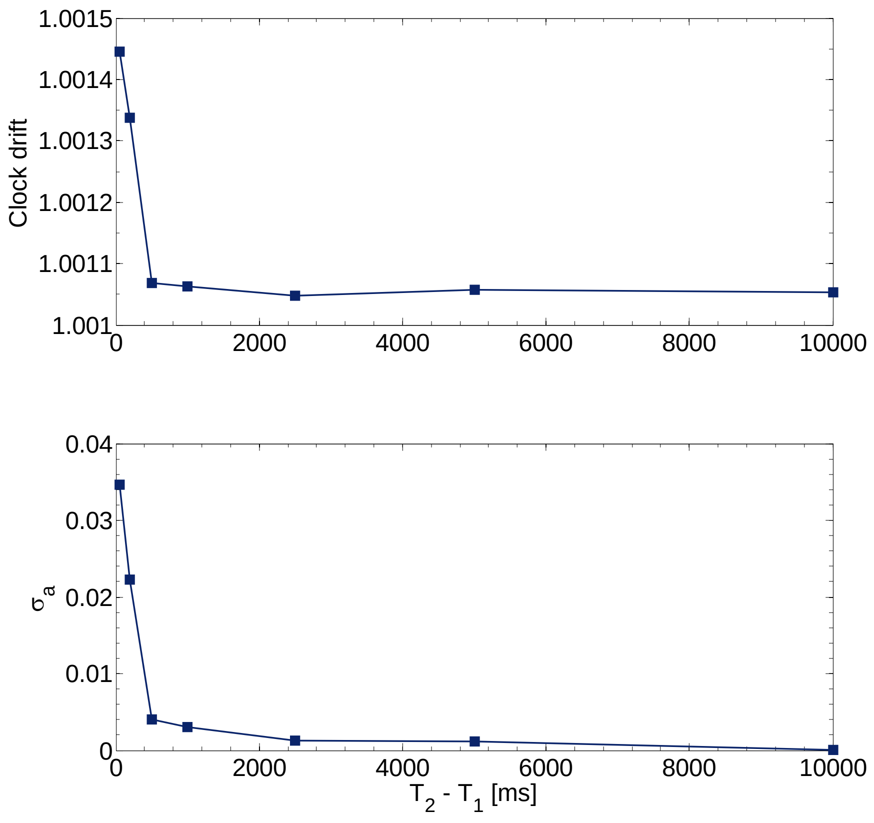

FIGURE 5: CLOCK DRIFT AVERAGE AND STANDARD DEVIATION FOR ONE MEASUREMENT FOR DIFFERENT VALUES OF T2-T1.

As regards to the variation of the estimated clock drift across different measurements, Figure 6 shows that as the time difference between T2 and T1 increases, the dispersion (errorbar) over the average value (red line) decreases significantly with values higher than 1000 ms (1 sec).

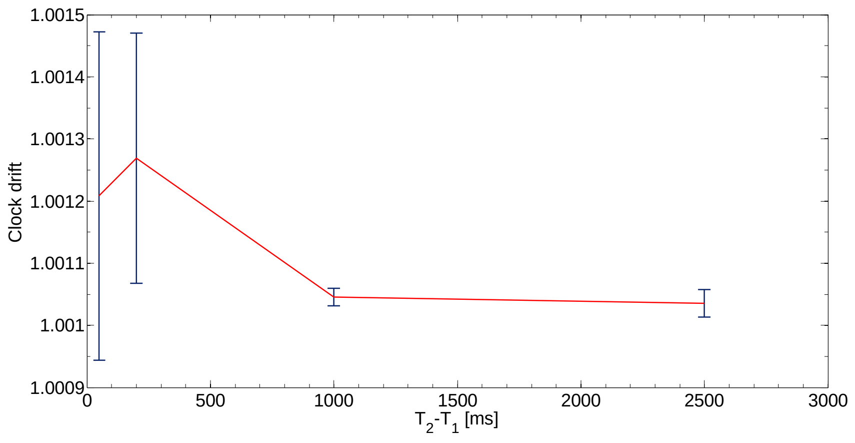

FIGURE 6: AVERAGE VALUE AND STANDARD DEVIATION (ERRORBAR) OF THE CLOCK DRIFT ACROSS DIFFERENT MEASUREMENTS AND FOR INCREASING VALUES OF T2-T1.

4.3 Clock offset

The offset has been estimated for different values of T4-T1 (refer to Figure 3) by repeating the same data exchange process several times during the same measurements (to have statistically significant values). However, it has to be noticed that what actually matters in this experiment is the time duration of the measurement itself, rather than the number of subsamples taken during the same measurement. This becomes clear in FIGURE 7, where the offset dispersion is shown for different values of T4-T1 (since the offset is expressed in epoch, the minimum value has been subtracted from the estimated offsets for the sake of visalization).

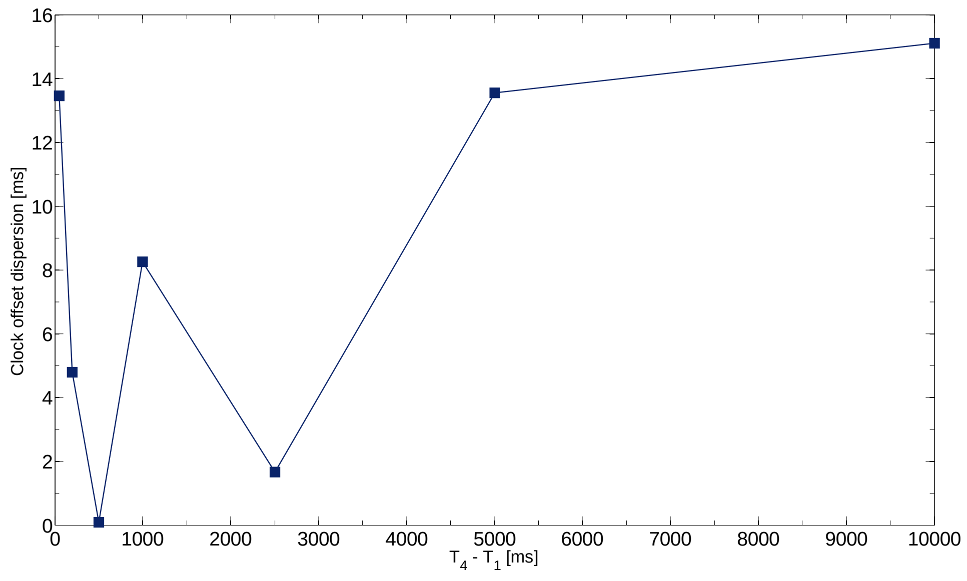

FIGURE 7: CLOCK OFFSET DISPERSION FOR DIFFERENT VALUES OF T4-T1.

On the basis of the estimated clock drift and clock offset, the synchronization error calculated on the data taken during the offset measurements, are depicted in Figure 8 and FIGURE 9, respectively for a sampling time equal to 50 ms and 5 seconds.

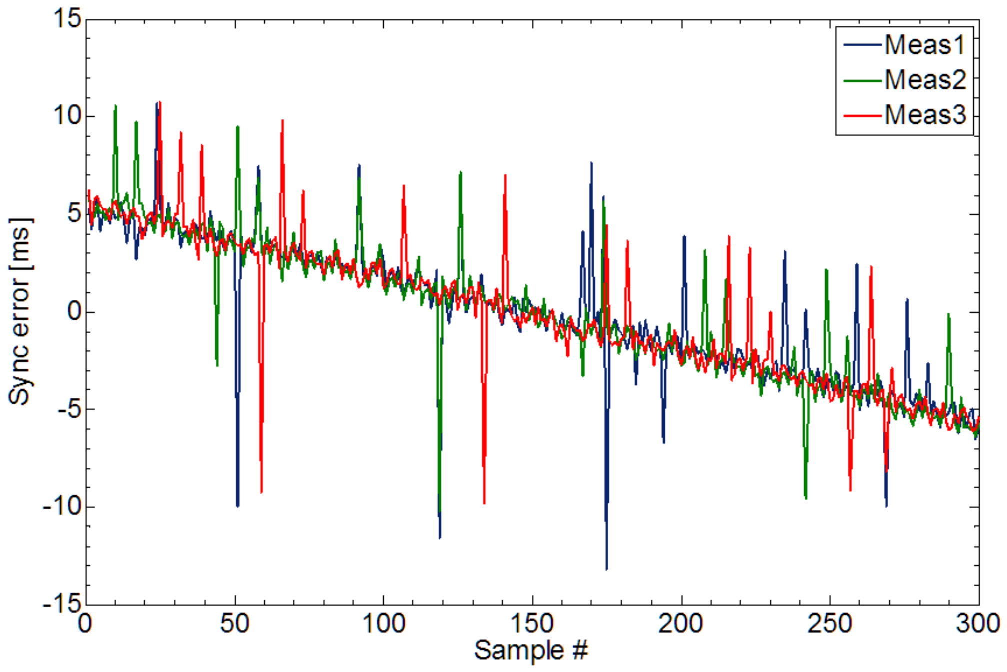

FIGURE 8: SYNC ERROR DURING THE OFFSET MEASUREMENT (T4-T1 = 50 MILLISECONDS) AFTER COMPENSATING FOR THE CLOCK DRIFT.

The effect of a wrong clock drift estimation is clear in Figure 8, where the slope error results actually in a straight line (error centered in 0 just for the sake of visualization). On the other hand, FIGURE 9 doesn't show a trend in the error, meaning that the clock estimation is close to the real value.

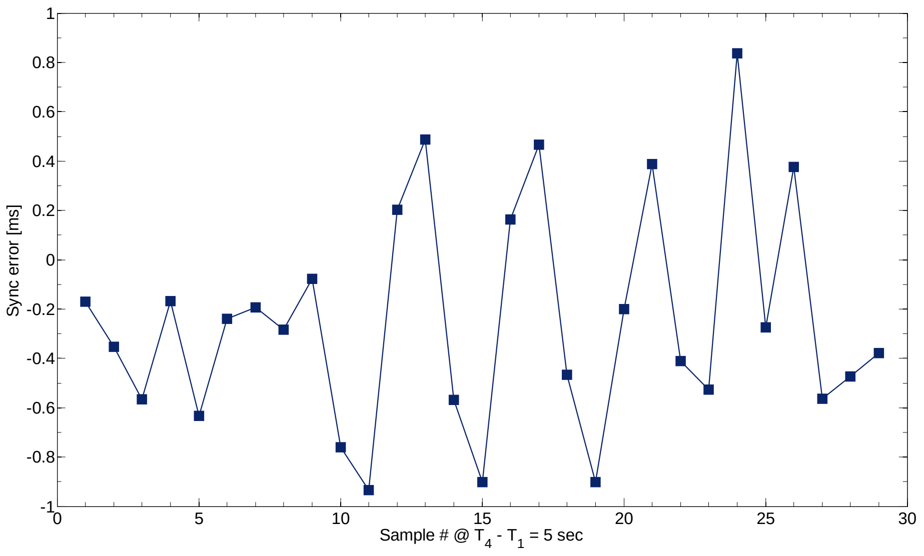

FIGURE 9: SYNC ERROR DURING THE OFFSET MEASUREMENT (T4-T1=5 SECONDS) AFTER COMPENSATING FOR THE CLOCK DRIFT.

4.4 Performance

The performance of the proposed method has been measured by computing the sync error after a relatively large amount of time (more than 1 hour) past after turning on one platform. Results are summarized in Figure 10 in terms of percentage error. As it can be noticed, the worst case, achieved with a sampling time of 50 ms, is about 0.02 % (the clock accuracy has improved roughly by a factor of 10), whereas the best case, achieved with a sampling time of 10 seconds (5 seconds and 2.5 seconds performs quite well all the same), is as low as 0.00064% (6 ppm).

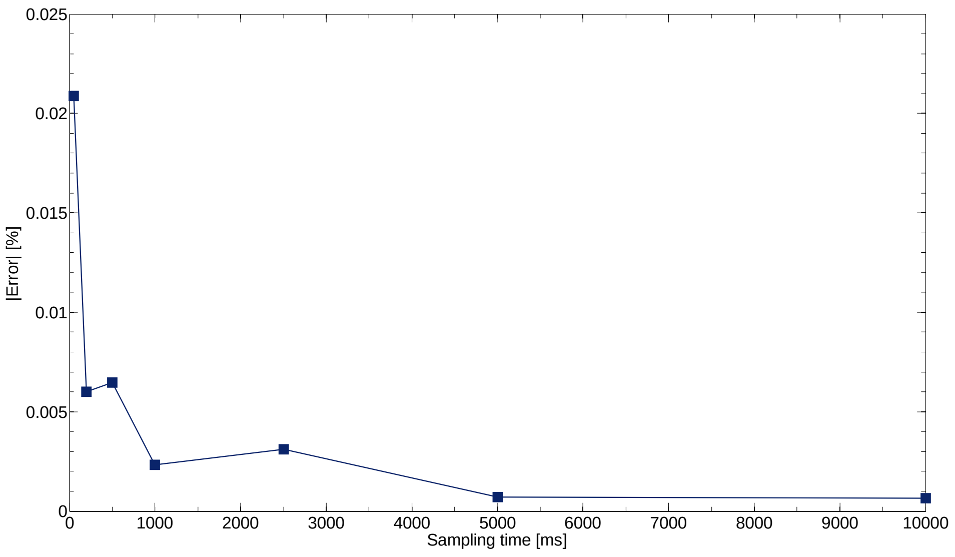

FIGURE 10: SYNC ERROR IN PERCENTAGE FOR DIFFERENT SAMPLING TIME VALUES

5 Conclusions and remarks

In this document, the basic principle of a synchronization method is outlined and assessed. Measurement results show that the sync error can be reduced down to about 10 ppm, which enables a time sync of acquisitions from different sensors with an accuracy of about 20 ms in half an hour of operation. This achievement comes at the price of dealing with the syncing task after (or before) the acquisition with a proper software procedure, which takes some time (tens of seconds per sensor) to be carried out.

6 References

[1] CSTCE8M00G55-R0 official page, http://www.murata.com/products/productdetail?partno=CSTCE8M00G55-R0.

[2] Fikret Sivrikaya, Bƒulent Yener, "Time Synchronization in Sensor Networks: A Survey".

[3] Network Time Protocol wiki page, https://en.wikipedia.org/wiki/Network_Time_Protocol

[4] M. L. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, New Orleans, LA, USA, 2003, pp. 1266-1273 vol.2. doi: 10.1109/WCNC.2003.1200555.

[5] António Grilo, "Wireless Sensor Networks lectures, Chapter 8: Time Synchronization", http://comp.ist.utl.pt/ece-wsn/doc/slides/sensys-ch8-Time-Synchronization.pdf