# ANALYSIS OF QUANTITIES INFLUENCING THE PERFORMANCE OF TIME SYNCHRONIZATION BASED ON LINEAR REGRESSION IN LOW COST WSNS

Domenico Capriglione,

Deborah Casinelli,

Luigi Ferrigno,

Università degli Studi di Cassino e del Lazio Meridionale,
Dipartimento di Ingegneria Elettrica e dell'Informazione,
Via G. Di Biasio 43, 03043, Cassino (FR) Italia
{capriglione, d.casinelli, ferrigno}@unicas.it

# ANALYSIS OF QUANTITIES INFLUENCING THE PERFORMANCE OF TIME SYNCHRONIZATION BASED ON LINEAR REGRESSION IN LOW COST WSNs

***Abstract.***

***Time synchronization in low-cost wireless sensor networks is a relevant issue for many researchers. In these networks, costs, energy consumption and indoor uses are relevant topics that generally lead to the adoption of synchronization algorithms rather than GPS-based synchronization devices. An even increasing number of papers dealing with the design and the implementation of protocol-based techniques for synchronization can be found in literature.***

***These synchronization algorithms can be divided into three main categories: one way messaging, two ways messaging and consensus based synchronization. Among the others, those based on the one way messaging and, in particular, on the adoption of regressive algorithms are widely used in many application contexts. Focusing the attention on this class of synchronization algorithms, this paper proposes a deep performance analysis aimed at highlighting the sensitivity of the regression-based algorithms to some factors that influence the accuracy in typical low cost applications, such as the finite resolution of the local timing clock, the presence of clock drift, clock high frequency noise and low frequency noise, the presence of latencies due to the radio devices, the presence of latencies due to the microcontroller device. The main goal is to evaluate the effect of each one of these factors of influence on the overall synchronization performance. To these aims, suitable analyses both in simulation environment and on real nodes have been carried out.***

***Keywords** Synchronization, wireless sensor network, measurement accuracy, measurement, regression*

# I. INTRODUCTION

Wireless Sensor Networks (WSNs) pervade many of today industrial and user applications. In most of these applications wireless sensors are small and low cost devices, with limited energy and computational resources that operate in small scale areas where no further infrastructure is present [1], [2] and have to "share" the same sense of time to obtain coordination among nodes, synchronized measurements, reliable data fusion, localization and environment monitoring to cite a few. Therefore the sensor network efficiency is largely dependent on the time synchronization among nodes [3]. Since the requirements of low-costs, low energy consumption and indoor use, this task is typically entrusted by synchronization protocols rather than specific hardware solutions (like Global Position System, GPS).

Recent literature is proposing many efficient algorithms for WSNs synchronization [4]-[6]. In the wide number of proposals present in literature, those based on linear regression are greatly explored by researchers since the promise of very good performance and energy consumption. These algorithms synchronize wireless sensor nodes by establishing a linear relationship between clocks of different sensor nodes with the aim of predicting a reference clock based on collected time-stamps.

The authors are involved in the topics of WSNs and synchronization from some years. In particular, they have focused the attention on the implementation and performance characterization of WSNs when low cost wireless sensor nodes are used. They have designed new wireless sensor nodes [7] and wireless interface architectures [8], as well as implemented and experimentally characterized the performance of synchronization schemes based on both two ways messaging and regression-based protocols [9]-[11].

Their studies evidenced that in typical industrial and user applications, where time synchronization is performed by devices that do not have specific hardware features, and in real scenario characterized by non-ideal clocks, real communication channels and processing devices, the performance can greatly change respect to one expected.

Focusing the attention on regression-based synchronization protocols, this paper extends previous studies performed in [11] and proposes a methodological approach for analyzing the above-mentioned causes affecting the synchronization performance. Typical factor of low cost applications, such as the finite resolution of the timing clock, the presence of clock drift, clock high frequency noise and low frequency noise, the presence of latencies due to both the radio devices and microcontroller device have been considered. The aim is to evaluate the effect of each one of these factors of influence on the overall synchronization performance. This study has been carried out performing a two-stage analysis. At first, a suitable simulation environment in which the factors of influence can be well controlled has been used. Then, considering real nodes, suitable set-up able to generate precise clock frequencies, to impose desired clock behaviors (i.e. variation of clock over time according the desired waveforms), to emulate fixed and variable latencies due to the communication channels, the serial tunneling and the application software has been adopted.

The final aim of the research is twofold. From one hand, the study will individuate techniques to mitigate the effect of these factors of influence and, from the other hand it could be useful for the definition of a reliable uncertainty model able to characterize the synchronization performance.

## II. MOTIVATION OF THE STUDY

Many papers have experienced that in real scenarios, synchronization protocols applied to real cheap wireless sensor nodes dramatically worsen their performance respect to that expected in theory [12]-[14].

Several factors of influence play a key role when real scenarios are involved. Among them, the most important concern with: (i) non-ideality of clocks that are characterized by drifts, high frequency noise and low frequency noise over time, (ii) the implementation on real low cost nodes (i.e. low cost microcontrollers without specific synchronization and time stamping features), (iii) the adoption of commercial radio systems with limited bandwidth and not negligible radio latencies, and (iv) the non-ideal behavior of the wireless channel.

In this framework, focusing the attention on the regression-based synchronization algorithms, it is expected that the number of collected time-stamps and the time interval among them could influence the overall performance.

As for the i) point, real low cost wireless sensor nodes generally adopt commercial clocks that exhibit non-ideal behavior characterized by drift, timing high frequency noise and low frequency noise. These behaviors are influenced by the clock accuracy and by other environmental quantities of influence as temperature, power supply stability, electromagnetic noise and so on. Regression-based synchronization algorithms are developed to estimate and compensate skews and offsets among clocks but are not designed to recover the effect of timing high frequency noise and low frequency noise. As a consequence, these factors of influence have to be taken into account for compensating also such effects by selecting the number of time-stamps to be adopted in the regression algorithm and the time intervals among them.

As for the ii) point, generally, low cost wireless sensor nodes do not feature an embedded radio-frequency (RF) device capable to generate time-stamps at Media Access Control (MAC) level: the microcontroller usually communicate with external RF devices through serial tunneling (e.g. TTL-232, I2C, SPI, etc). These serial tunnels introduce additive latencies in communication that could worsen the overall performance of the synchronization protocols. In particular, these latencies can be divided into fixed and variable latencies. The former are due to the baud rate of the serial tunneling and to the number of byte transmitted in the message, while the latter are due to possible conflict in the serial bus, retransmissions, and time variability of the involved devices [15]-[17]. Even though the former can be approximately estimated and compensated, the latter are variable, unwanted and uncontrollable delays that worsen the transmission process.

Furthermore, real low cost nodes adopt a finite resolution in the time measurement routines. This resolution adds a further worsening effect in the overall synchronization performance.

As for the iii) point, typical radio systems adopted in WSNs as the WI-FI, BT and ZigBee to cite a few, show variable latencies that depend on the considered radio system, the considered radio device, the number of devices present in the network and the network state [18]-[24]. Also in this case their effects have to be adequately investigated and suitable mitigating techniques should be designed and employed [25], [26] when suitable time-stamping capabilities are not available.

As for the wireless channel (iv), the distance among nodes, the mobility of nodes, the presence of radio interferences, the presence of multipath cause a number of messages to be retransmitted. This can be seen as an additional variable, unwanted and uncontrolled latency which effect has to be adequately explored.

Taking into account the previous considerations, in the following, after a brief theoretical background about the regression-based synchronization algorithm and the timing model of a synchronization procedure, the main causes of influence are analyzed in detail and their effects are evaluated in a suitable simulation environment.

Finally, some tests carried out in a real scenario are presented to assess the obtained results. In particular, a suitable set-up able to generate precise clock frequencies, to impose desired clock behaviors (i.e. variation of clock over time according the desired waveforms), to emulate fixed and variable latencies due to the communication channels, the serial tunneling and the application software is adopted.

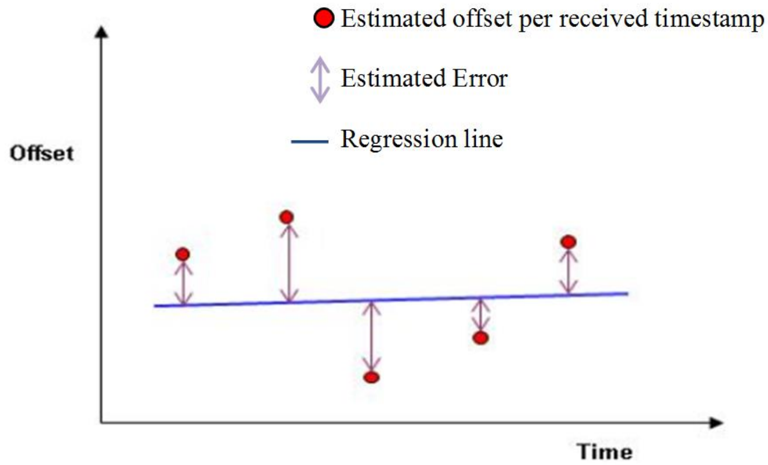


Figure 1 Operating of a regression-based synchronization algorithm.

# III. THEORETICAL BACKGROUND

## A. Notes on the linear regression-based synchronization

Aim of this class of synchronization methods is to achieve network wide synchronization between the local clocks of the participating nodes. Often these algorithms provide multi-hop synchronization in which the root of the network manages and maintain the global time. All the other nodes synchronize their clocks to the one of the root.

The operating of a regression-based synchronization algorithm can be described as follows.

The sender sends a message that contains its own time-stamp. Receivers use the received time-stamp to generate their initial local clocks. Then the sender dispatches a number of time-stamps generally with regular time intervals. Each receiver compares the received time-stamp with that obtained from its local clock at the receiving time. If the corresponding drifts are linear (see Figure 1), a regressive pattern may be predicted and used to compensate errors, thus reducing the updating rate. This class of synchronization protocols has the advantages of having low transmission loss and being robust at the temporary or persistent unavailability of some nodes, but it has the disadvantage of collecting more time-stamps to put into effect a linear regression.

**B. Timing model of time-stamp delay in low cost wireless sensor nodes**

Figure 2 shows an example of data communication between two wireless sensor nodes, namely one sender and one receiver, during a synchronization procedure. In particular, Figure 2a) shows the simplified architecture of two low cost nodes involved in the data communication where, for the sake of clarity, the sensor nodes have been represented through microcontrollers and radio systems.

Figure 2b) shows the timing model of the generation, sending, propagation and reception of time-stamps. Figure 2b) extends the typical timing model proposed in [6] and generally adopted in the analysis of synchronization performance when timing at MAC level is available. In particular, the proposed model adds the “Microcontroller side latency” which accounts for the following main factors:

- the latencies due to the communication between the microcontrollers and the radio devices of Figure 2a). In fact, these devices are usually connected by a serial tunnel as the TTL-232, the I2C, the SPI, and so on;

- the non-idealities of the clocks, related to drifts and low and high frequency noise;

- the latencies related to the time-stamp forming, due to the delay of execution of the Interrupt Service Routine and to timer resolution adopted for the time-stamp evaluation.

More in detail, the microcontroller performs the synchronization procedure by generating the required time-stamps. These time-stamps are typically generated through interrupt driven software routines able to interrogate the system timers. At this stage a variable latency is introduced as dependent on the current processing of the microcontroller that might have access to these routines after some delay. Indeed, it is here reminded that the synchronization procedure is not the primary task of a wireless sensor node, but it is usually a service routine that allows to improve (or warrant) the reliability of a main task. Furthermore, the generated time-stamp has a finite resolution bounded to the

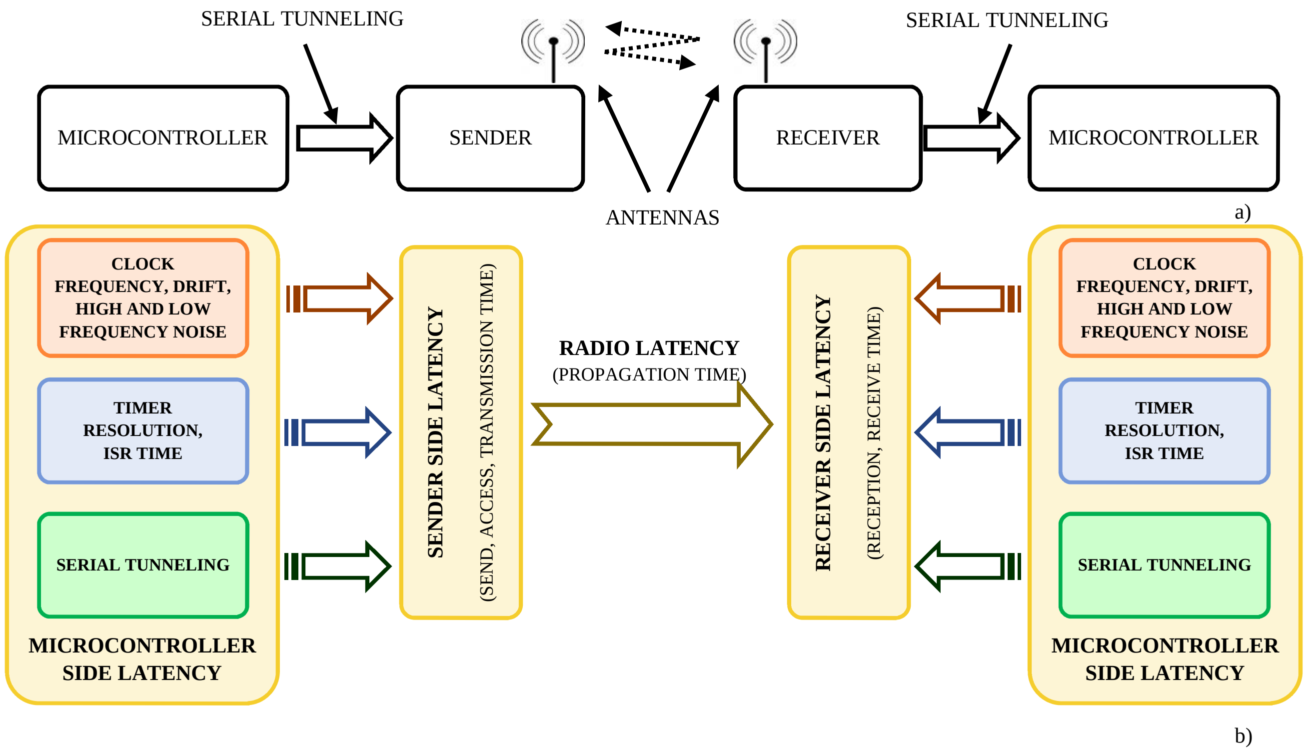


Figure 2 Simplified diagram of communication among low cost wireless sensor nodes (a), and, highlight of the timing influence quantities during the time-stamps creation, transmission and reception phases (b).

prescaler value used for the time evaluation in the timer register. All these latencies might be considered as “microcontroller side latency”.

Subsequently, following the scheme of Figure 2a), the generated time-stamps are sent, through the serial link, to the radio device. This stage introduces two additive latencies. The former depends on the baud rate of the serial channel and the number of transmitted bytes. Generally, the mean value of this latency can be corrected and it is not considered in the following analysis. In this way only the effect of the clock drift and resolution affects this kind of latency. The latter depends on the adopted wired link between the microcontroller and the radio device and it is related to the imposed packet timing, the number of conflicts, the number of retransmissions and so on. These last latencies have been indicated in Figure 2b) in the “serial tunneling”block.

Then, the time-stamps are processed by the radio device. At this stage other latencies are introduced (namely, the SENDER SIDE LATENCY block of figure 2b). In particular: i) the send time is a random time spent for the generation of the time-stamps from the application to the MAC level of the radio device; ii) the access time is a random time spent for the access to the wireless radio channel. This time strongly depends on the adopted radio standard. As an example, if seven wireless sensor nodes are involved, the Bluetooth radio standard is adopted and a data rate DM5 is involved, each node might access to the channel after a time varying from 625 us up to 22.5 ms [18]; iii) the transmission time is the time spent for the data transmission at the physical layer.

After, a propagation time spent for the propagation on the wireless physical link completes the model at the sender side (named RADIO LATENCY in figure 2b).

At the receiver end, the time-stamps reach the receiver radio-frequency device. Here the reception and receive times are introduced. The former is the time spent to receive time-stamps and to send them from the physical to the MAC layer, the latter is the time spent to send the packet to the application layer of the radio device. Finally, as for the microcontroller side, latencies similar to those introduced at the sender side can be considered.

**C. The clock model**

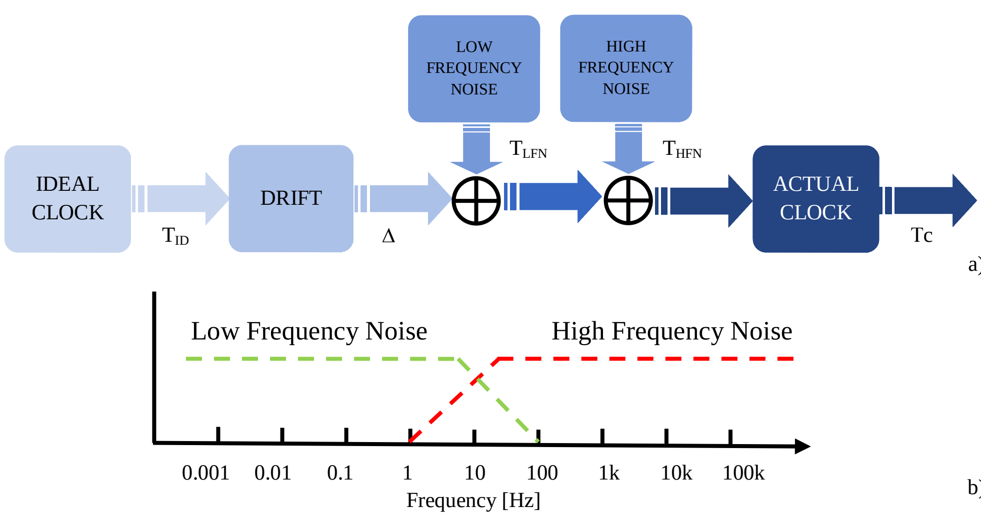


Figure 3 Simplified model of a real clock. a): general scheme, b): low frequency noise and high frequency noise areas of influence.

In this paper the simplified clock model considered by ITU-T G.810 is applied [27]. In particular, the clock ($T_C$) of a sensor node can be expressed as:

$$T_C = \Delta + T_{HFN}(t) + T_{LFN}(t) \quad (1)$$

Where Δ is the value of the clock after the application of the drift, $T_{HFN}(t)$ is the high frequency noise component and $T_{LFN}(t)$ is the low frequency noise component (Figure 3a). ITU-T G.810 defines the timing high frequency noise as “the short-term variations of the significant instants of a timing signal from their ideal positions in time (where short-term implies that these variations are of frequency greater than or equal to 10 Hz)”. The low frequency noise is defined as “the long-term variations of the significant instants of a digital signal from their ideal position in time (where long-term implies that these variations are of frequency less than 10 Hz)” (Figure 3b).

Adopting the time model of equation 1, and considering a sender-receiver linear regression-based synchronization procedure, it is possible to express the receiver clock ($T_R$) in function of the sender clock ($T_C$) by the following relation:

$$T_R = (\alpha T_C + \beta) + T_{HFN\_R}(t) + T_{LFN\_R}(t) \quad (2)$$

Where α and β are respectively the skew and the offset of the receiver clock with respect to the sender clock ($T_C$), $T_{HFN\_R}(t)$ and $T_{LFN\_R}(t)$ are respectively the overall high frequency noise and low frequency noise components on the receiver clock.

## IV. Setup of the Experiments

### A. Preliminary considerations

In order to analyze the causes affecting the synchronization performance, according to the timing model and the clock model shown in Section III, a software, developed in Matlab environment, for simulating and analyzing the performance of a linear regression-based algorithm [28] has been

Table I Parameters considered for the simulation

| Parameter | Description | Value |
|---|---|---|
| $f_0$ | Clock frequency | 8 MHz |
| Baud Rate | Serial data speed | 38400 bps |
| TIMER PRESCALER | Sets the microcontroller timer finite resolution | 1, 32, 256 |
| ISR Time | Average value [29] | 8 µs |
| Serial tunneling | Average value (obtained by measurement campaigns) | 18 µs |
| $T_S$ | Time interval between two consecutive synchronizations rounded to the microcontroller resolution. | 36 ms, 360 ms, 3.6 s, 36 s, 360 s, 3600 s, 36000 s, 72000 s |
| DRIFT | Frequency drift between sender and receiver clocks | 10, 100, 1000 ppm |
| HFN | Percentage high frequency noise variations | 0, 0.025, 0.05% ppm |
| LFN | Percentage low frequency noise variations | 0 %, 0.1 %, 0.2 % |
| N_REGR | Number of requested time-stamp to apply the linear regression | 8 |
| RADIO LATENCY: sum of | Minimum value [22]-[24] | Bluetooth: 7 ms |
| | | WI-FI: TCP 20 ms<br>UDP 1.78 ms |
| | | ZigBee: 15 ms |
| | Maximum random value, bound to packet retransmission [18]-[20]. | Bluetooth: 1.27937 s |
| | | WI-FI: 27 ms |
| | | ZigBee: 26.04 ms |

designed. Then, the obtained results (reported in section V) have been validated by considering real nodes and the experimental setup described in section VI.

The basic idea is to set up a test environment able to analyze the effects of the considered parameters of influence and of their possible combinations on the synchronization performance. In particular, all the parameters highlighted in section III have been considered to take into account the effects of the clocks non-idealities, the propagation time, the microcontroller side latency, the serial tunneling latency [15]-[17], the minimum latency provided by the standard of the radio device in use [22]-[24], and the latency caused by packet retransmission [18]-[20] (in order to simulate even the effects of electromagnetic interference and other causes of packet loss in the radio channel).

In particular, the variations of the following parameters have been considered in the simulation environment: (i) the synchronization interval, (ii) the timer resolution, (iii) the amplitude of random timing due to high frequency noise and low frequency noise, (iv) the frequency drift between the sender and the receiver clocks and (v) the random contribution of the radio latency. The number of time-stamps adopted in the linear regression has been set at 8, as in the FTSP synchronization protocol [6].

The range of variation of the clock parameters (drift, high frequency noise and low frequency noise) has been derived from an experimental campaign on a number of real clocks [9]. The clock resolution has been chosen by considering typical microcontroller prescaler values. The communication latencies of radio devices have been directly derived by the related standards. In the whole test the length of the timing message that contains the time-stamp and the baud rate adopted in the microcontroller to radio device communication has been fixed to a unique value (38400 bps). This choice not leads to a lack of generality since these parameters cause a known fixed latency, due to the microcontroller to radio device communication that is compensated by the regression. Finally, to reduce the number of tests, also the low frequency noise period, of both sender and receiver clocks, has been fixed to one day. The whole considered parameters and their values are listed in Table I.

**B. The implemented software**

In the implemented software [28], the typical communication procedure between two nodes to be synchronized each other, one sender and one receiver has been simulated according to the model of Figure 2. The sender transmits its time-stamp to the receiver, which can synchronize its time reference using an algorithm based on linear regression: the first time-stamp sent by the sender is stored by the receiver and used as its reference for the subsequent time-stamps associated to the receiver clock [6]. Those time-stamps take into account the main sources of synchronization error defined as the time difference between the sender and the receiver time-stamps. In particular, on the sender side, the time-stamp value takes into account the clock high frequency noise and low frequency noise, as defined in [27], the finite resolution of the microcontroller timer and the interval between two successive time-stamps.

With reference to figure 2b), the received time-stamp is, then, obtained by adding to the sent time-stamp the time contributions due to the sender microcontroller side latency, the sender side latency, the radio latency, the receiver side latency and receiver microcontroller side latency.

Time-stamps collected on receiver side, are used to obtain the optimal estimators of the standard linear regression (α and β). Then, the receiver clock is corrected according to the way sketched in Figure 1 [6]. After such a correction, the time-stamps collected on the sender and receiver sides are adopted to measure the residual error present after also the synchronization procedure. In the following, this quantity is referred as “synchronization error”.

## V. NUMERICAL RESULTS

The software described in the previous section has run for evaluating the performance of the linear regression-based algorithm, under different test conditions achieved by varying the parameters of influence in the ranges of Table I. In particular, for each test condition, the performance is evaluated calculating the synchronization error as average value on 100 synchronization procedures (performed

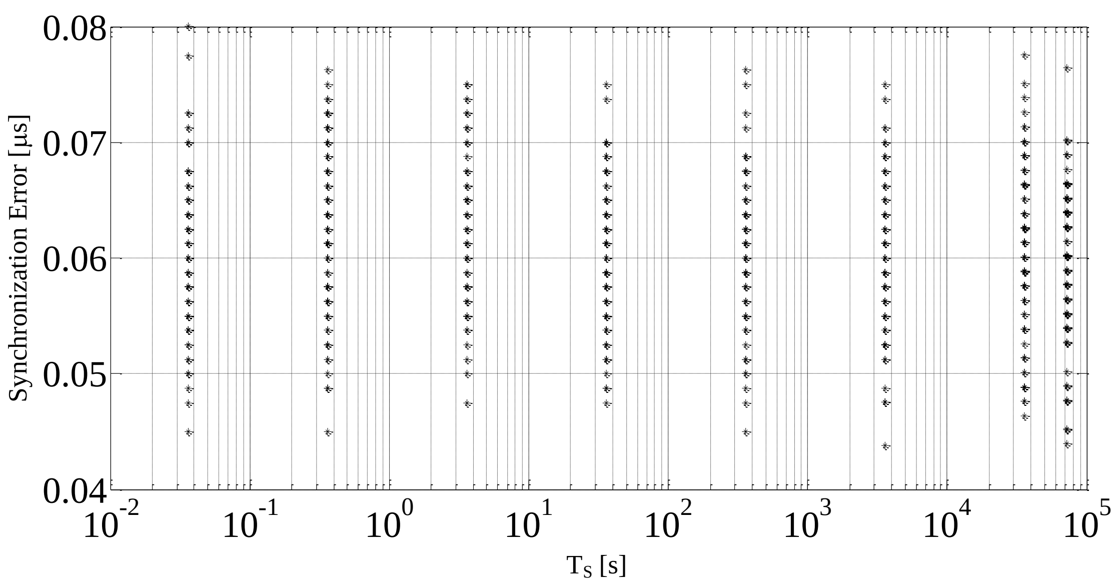


Figure 4 Synchronization error vs $T_S$ when ideal clocks, microcontroller prescaler set to 1 and no radio latencies are involved.

considering a time interval between two consecutive synchronizations equal to the selected $T_S$), and each synchronization procedure has been repeated 100 times (for a total of 10000 tests).

**A. The ideal case**

In this analysis the sender and receiver clocks are considered ideals (i.e. neither drift or high frequency noise nor low frequency noise are applied), the microcontroller prescaler is set to 1 (this means that the period of the timer register tick is 125 ns when the clock frequency is 8 MHz and the High speed PLL mode is used), in order to have a precise clock, as well as no radio latency is considered. In this way the relationship between the average synchronization error and the time interval between two consecutive synchronizations, $T_S$, in absence of other non-idealities, has been studied. Figure 4 shows the evolution of the synchronization error versus $T_S$. It evidences that the average synchronization error, in case of ideal clocks, is not influenced by the synchronization interval and lies around an average value of 60 ns. This error that should have been ideally equal to zero is due to the considered propagation time and time quantization due to the considered prescaler value. As previously

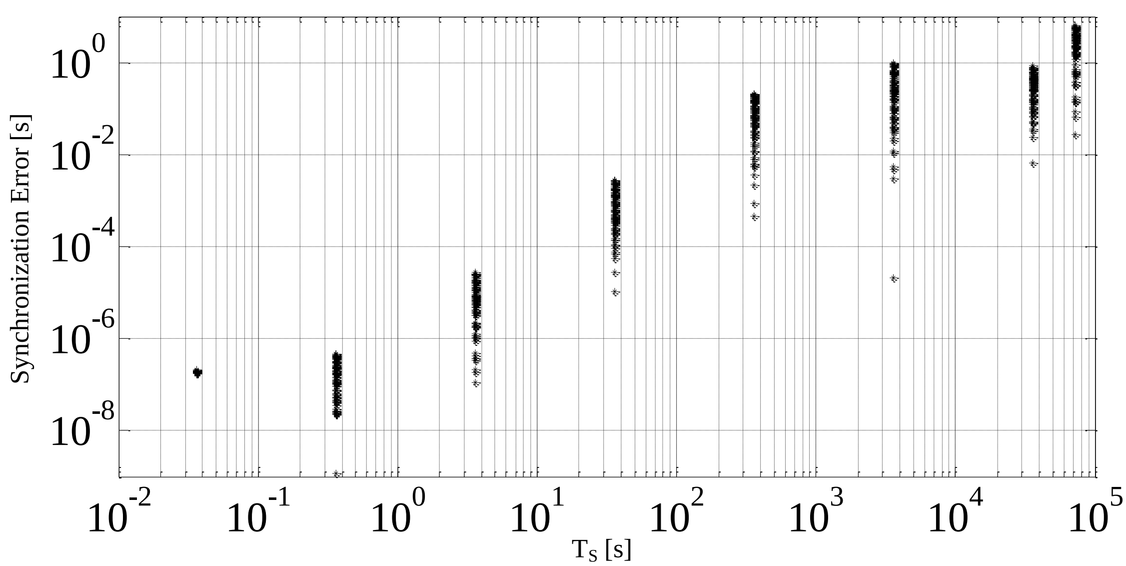


Figure 5 Synchronization error vs Ts, in the case of non-ideal clocks (presence of high and low frequency noise and drift phenomena considered at their maximum values of Table I) and microcontroller prescaler set to 1. No radio latencies are involved.

said, for each of the $T_S$ values, the synchronization procedure has been repeated for 100 times.

### B. Effects of the clock non-idealities

In this analysis, the effect of drift, high frequency noise and low frequency noise of the sender and receiver clocks when they assume the maximum values reported in Table I, have been investigated.

In Figure 5 the average synchronization errors is reported. It can be noticed that the average synchronization error arises, with respect of the results reported in Figure 4, of several orders of magnitude, highlighting the combined effect of the synchronization interval and the clocks non idealities.

### C. Effects of the radio latency

In this analysis, we have investigated the effect of radio latency when ideal clocks are considered, for different commercial radio standard in use. Their random contribution to the latency has been considered to define the variability range of such random quantity. by considering a rectangular distribution in the range specified in Table I.

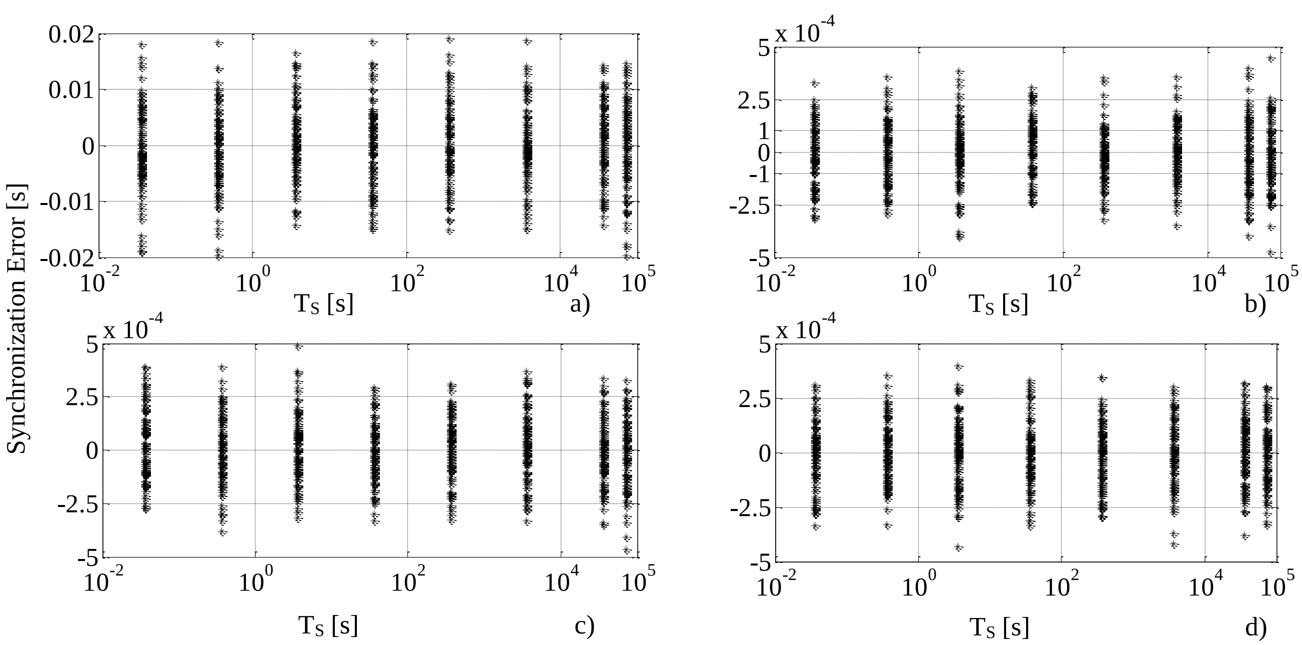


Figure 6 Synchronization error vs $T_S$, in case of ideal clocks, with prescaler set to 1, when radio latencies are involved: a) Bluetooth; b) WI-FI TCP; c) WI-FI UDP; d) ZigBee.

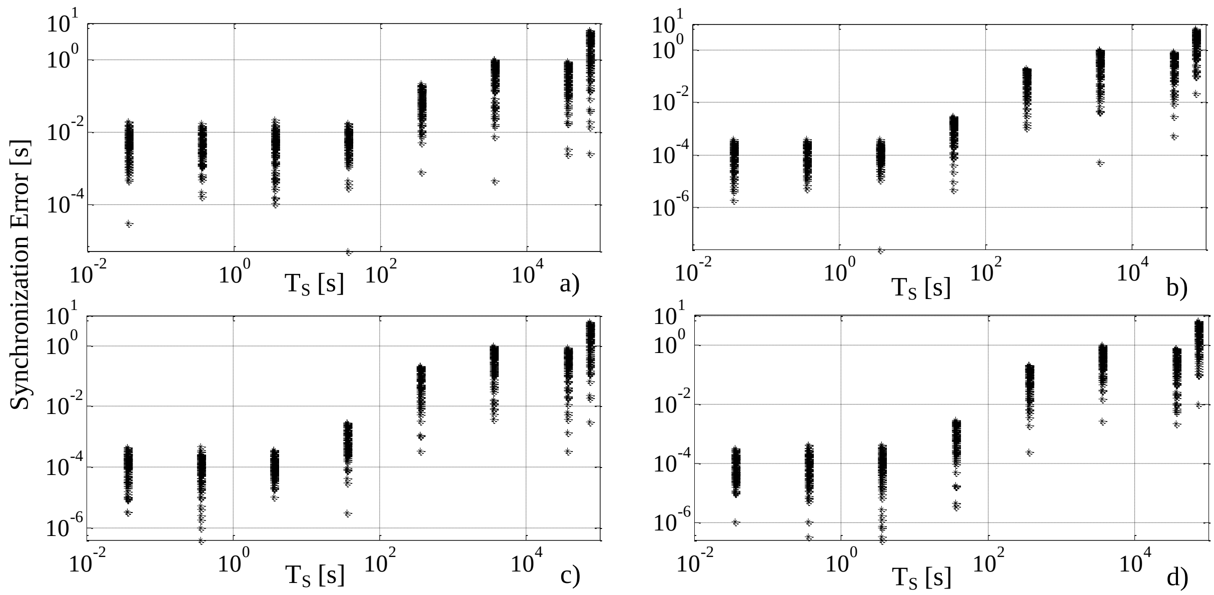


Figure 7 Synchronization error in the case of wireless latencies (presence of high and low frequency noise and drift phenomena considered at maximum values of Table I) when radio latencies are involved: a) Bluetooth; b) WI-FI TCP; c) WI-FI UDP; d) ZigBee

Figure 6 shows the obtained results: the average synchronization error has the same order of magnitude of the case study when no radio latency is considered (see Figure 4), even if in this case, a larger dispersion is observable also for low values of $T_S$. The average synchronization error is almost constant with $T_S$: this means that when an ideal clock is involved, the radio latency affects the synchronization error in the same way for each synchronization interval.

The synchronization error is influenced by the considered radio standard. In particular the higher the radio latency value the worse the synchronization error. For example, the Bluetooth has the greatest radio latency (Table I) and its overall average synchronization error is the highest with respect to the other communication standards.

**D. Combined effects of the radio latencies and clock non-idealities**

Figure 7 shows the combined effect of the radio latencies and of the non-idealities of clocks, when all the parameters that concern the clocks are set at their maximum value.

It is evidenced that the average synchronization error increases with the synchronization interval, differently from the previous case where ideal clocks where considered. In order to highlight the weight of each parameter with respect to the synchronization error, a deeper analysis has been performed by considering separately, one at time, the clock non-idealities (prescaler, drift, low frequency noise and high frequency noise, respectively).

In particular, for each test condition, the performance is evaluated calculating the synchronization error evaluated as average value on 100 synchronization procedures. Each synchronization procedure has been repeated 100 times and then $\mu_E$, and the corresponding mean standard deviation, $\sigma_{\mu E}$ have been evaluated. The clock sender model has always been considered ideal in the further analysis.

Table°II shows the effect of the prescaler when neither drift nor high frequency noise nor low

Table II Synchronization error vs Prescaler values

| | | | NO LATENCY | | BLUETOOTH | | WI-FI TCP | | WI-FI UDP | | ZIGBEE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | PS | $\mu_E$ [µs] | $\sigma_{\mu E}$ [ns] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] |
| $T_S$ [s] | 0.036 | 1 | 0.19 | 0.69 | -0.88 | 0.73 | -0.020 | 0.014 | 0.030 | 0.017 | -0.001 | 0.014 |
| | | 32 | 1.98 | 10.29 | 0.004 | 0.72 | -0.034 | 0.014 | 0.003 | 0.016 | 0.012 | 0.015 |
| | | 256 | 15.20 | 67.54 | 0.22 | 0.66 | 0.025 | 0.014 | -0.010 | 0.015 | 0.032 | 0.014 |
| | 0.36 | 1 | 0.19 | 0.69 | -0.89 | 0.69 | -4.09E-04 | 0.014 | 0.008 | 0.016 | 0.006 | 0.013 |
| | | 32 | 2.26 | 7.69 | -0.96 | 0.70 | 0.003 | 0.016 | -0.009 | 0.015 | -0.012 | 0.014 |
| | | 256 | 17.07 | 71.51 | -0.29 | 0.67 | 0.025 | 0.014 | 0.033 | 0.015 | 0.003 | 0.014 |
| | 3.6 | 1 | 0.19 | 0.61 | 0.67 | 0.70 | -0.004 | 0.015 | 0.016 | 0.016 | 0.000 | 0.016 |
| | | 32 | 2.23 | 8.49 | 1.33 | 0.74 | -0.034 | 0.015 | -0.001 | 0.017 | 0.012 | 0.014 |
| | | 256 | 17.11 | 60.58 | 0.80 | 0.69 | 0.004 | 0.016 | -0.003 | 0.015 | 0.034 | 0.015 |
| | 36 | 1 | 0.19 | 0.69 | -0.19 | 0.72 | 0.026 | 0.014 | 0.005 | 0.014 | -0.016 | 0.015 |
| | | 32 | 1.99 | 9.61 | -0.56 | 0.77 | 0.019 | 0.014 | -0.004 | 0.015 | 0.024 | 0.014 |
| | | 256 | 14.98 | 71.58 | 1.10 | 0.71 | 0.018 | 0.014 | 0.022 | 0.014 | 0.026 | 0.014 |
| | 360 | 1 | 0.19 | 0.59 | 0.30 | 0.71 | -0.021 | 0.013 | 0.008 | 0.014 | -0.014 | 0.014 |
| | | 32 | 1.98 | 9.21 | 1.03 | 0.68 | -0.008 | 0.014 | 0.009 | 0.013 | -0.009 | 0.014 |
| | | 256 | 14.92 | 70.43 | 0.40 | 0.67 | 0.029 | 0.015 | -0.013 | 0.015 | 0.022 | 0.014 |
| | 3600 | 1 | 0.19 | 0.77 | -0.54 | 0.67 | -0.008 | 0.013 | 0.019 | 0.016 | -0.003 | 0.014 |
| | | 32 | 2.05 | 9.18 | 0.15 | 0.70 | -0.005 | 0.013 | -0.012 | 0.015 | 0.001 | 0.013 |
| | | 256 | 15.23 | 56.66 | 0.34 | 0.72 | 0.022 | 0.014 | -0.002 | 0.015 | 0.020 | 0.013 |
| | 36000 | 1 | 0.19 | 0.63 | 1.17 | 0.67 | -0.029 | 0.017 | -0.017 | 0.015 | 0.000 | 0.015 |
| | | 32 | 1.98 | 9.33 | 0.07 | 0.65 | -0.008 | 0.015 | -0.019 | 0.015 | 0.022 | 0.016 |
| | | 256 | 14.96 | 77.37 | -0.90 | 0.75 | 0.009 | 0.015 | 0.016 | 0.013 | 0.013 | 0.013 |
| | 72000 | 1 | 0.19 | 0.60 | 0.12 | 0.76 | -0.021 | 0.016 | -0.008 | 0.016 | -0.008 | 0.015 |
| | | 32 | 2.00 | 9.60 | -0.93 | 0.74 | -0.018 | 0.015 | -0.015 | 0.016 | -0.016 | 0.014 |
| | | 256 | 14.93 | 74.21 | 1.16 | 0.67 | 0.008 | 0.015 | 0.006 | 0.013 | 0.039 | 0.014 |

frequency noise affect the clocks behavior, with and without radio latencies applied. The $\mu_E$ value is constant with the prescaler and it is only influenced by the value of the applied radio latency. The $\sigma_{\mu E}$ value is, instead, comparable with the average synchronization error. When no radio latency is applied, both $\mu_E$ and $\sigma_{\mu E}$ vary with the prescaler. With reference to $T_S$ the obtained results show that the absolute value of the synchronization error generally increases with $T_S$.

In Table°III the error is evaluated by varying the percentage value of the receiver low frequency noise, when there is neither drift nor high frequency noise between sender clock and receiver clock but the lowest prescaler is set and radio latency is applied. In this case, both $\mu_E$ and $\sigma_{\mu E}$ increase with the synchronization interval and with the low frequency noise, whatever be the radio system involved.

Table III Synchronization Error vs Low Frequency Noise

| | | | NO LATENCY | | BLUETOOTH | | WI-FI TCP | | WI-FI UDP | | ZIGBEE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | LFN [%] | $\mu_E$ [μs] | $\sigma_{\mu E}$ [ns] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] |
| $T_S$ [s] | 0.036 | 0.10 | 0.19 | 6.7 | 0.23 | 0.61 | 0.011 | 0.014 | -0.010 | 0.016 | 0.005 | 0.014 |
| | | 0.20 | 0.19 | 6.7 | 0.30 | 0.61 | 0.013 | 0.015 | -0.014 | 0.016 | 0.009 | 0.014 |
| | 0.36 | 0.10 | 0.19 | 48.4 | 1.21 | 0.78 | -0.008 | 0.014 | 0.013 | 0.015 | 0.011 | 0.014 |
| | | 0.20 | 0.18 | 102 | 1.30 | 0.66 | -0.013 | 0.014 | 0.006 | 0.014 | -0.010 | 0.015 |
| | 3.6 | 0.10 | -0.01 | 4.7E+03 | -1.9 | 0.68 | -0.002 | 0.016 | 0.028 | 0.015 | 0.018 | 0.015 |
| | | 0.20 | 1.5 | 9.6E+03 | 0.77 | 0.70 | 0.003 | 0.015 | -0.009 | 0.016 | 0.017 | 0.015 |
| | 36 | 0.10 | 11.5 | 5.3E+05 | 0.39 | 0.78 | -0.052 | 0.051 | 0.014 | 0.054 | 0.12 | 0.046 |
| | | 0.20 | 102 | 1.0E+06 | 1.3 | 0.63 | -0.13 | 0.10 | 0.12 | 0.10 | 0.024 | 0.094 |
| | 360 | 0.10 | 4.1E+03 | 3.5E+07 | 0.69 | 3.9 | 5.4 | 3.6 | -0.1 | 3.9 | 6.2 | 3.7 |
| | | 0.20 | 4.8E+03 | 6.7E+07 | 4.7 | 7.3 | 2.6 | 7.5 | 5.4 | 7.1 | 1.5 | 7.8 |
| | 3600 | 0.10 | 1.1E+04 | 1.8E+08 | -4.8 | 17.3 | -2.1 | 18.0 | 7.1 | 17.3 | -20.9 | 17.8 |
| | | 0.20 | 4.1E+04 | 3.4E+08 | 32.6 | 35.0 | 31.7 | 34.3 | 48.7 | 34.7 | 22.1 | 34.9 |
| | 36000 | 0.10 | -3.3E+03 | 1.5E+08 | -18.8 | 14.3 | 1.7 | 15.3 | -17.1 | 15.5 | -31.0 | 14.2 |
| | | 0.20 | -2.5E+04 | 2.9E+08 | -12.0 | 28.4 | 3.6 | 29.4 | -6.3 | 28.9 | 40.3 | 29.5 |
| | 72000 | 0.10 | -1.5E+04 | 1.1E+09 | 13.9 | 107 | -62.0 | 116 | 120 | 104 | -142 | 113 |
| | | 0.20 | -1.6E+04 | 2.2E+09 | -278 | 231 | -128 | 232 | -188 | 222 | 57.6 | 223 |

Table IV Synchronization Error vs High Frequency Noise

| | | | NO LATENCY | | BLUETOOTH | | WI-FI TCP | | WI-FI UDP | | ZIGBEE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | HFN [ppm] | $\mu_E$ [μs] | $\sigma_{\mu E}$ [ns] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] |
| $T_S$ [s] | 0.036 | 0.025 | 0.19 | 0.68 | -0.62 | 0.71 | 0.001 | 0.015 | -0.034 | 0.015 | 0.024 | 0.014 |
| | | 0.05 | 0.19 | 0.72 | -1.1 | 0.76 | -0.043 | 0.016 | -0.025 | 0.016 | 0.001 | 0.013 |
| | 0.36 | 0.025 | 0.19 | 0.71 | -1.00 | 0.68 | -0.005 | 0.015 | -0.014 | 0.014 | 0.013 | 0.015 |
| | | 0.05 | 0.19 | 0.66 | -0.25 | 0.73 | -0.007 | 0.015 | -0.020 | 0.015 | 0.010 | 0.014 |
| | 3.6 | 0.025 | 0.19 | 1.3 | 0.06 | 0.67 | -0.018 | 0.017 | -0.006 | 0.014 | 0.013 | 0.014 |
| | | 0.05 | 0.19 | 2.5 | 0.10 | 0.74 | 0.021 | 0.016 | 0.015 | 0.015 | -0.027 | 0.014 |
| | 36 | 0.025 | 0.21 | 11.0 | -0.11 | 0.75 | -0.007 | 0.015 | -0.003 | 0.015 | -0.009 | 0.016 |
| | | 0.05 | 0.20 | 25.4 | 0.47 | 0.72 | -0.040 | 0.014 | 0.011 | 0.015 | -0.035 | 0.013 |
| | 360 | 0.025 | 0.33 | 122 | -0.15 | 0.70 | 0.009 | 0.014 | 0.007 | 0.016 | -0.020 | 0.013 |
| | | 0.05 | 0.33 | 237 | 0.42 | 0.64 | 0.032 | 0.015 | 0.013 | 0.016 | -0.003 | 0.014 |
| | 3600 | 0.025 | -0.35 | 1.1E+03 | -0.77 | 0.63 | -0.003 | 0.014 | -0.022 | 0.016 | 0.022 | 0.013 |
| | | 0.05 | 3.7 | 2.4E+03 | -0.54 | 0.77 | 0.006 | 0.016 | 0.035 | 0.016 | 0.005 | 0.015 |
| | 36000 | 0.025 | -12.0 | 1.1E+04 | -1.2 | 0.75 | 0.018 | 0.018 | -0.013 | 0.019 | -0.014 | 0.016 |
| | | 0.05 | -7.7 | 2.3E+04 | 0.06 | 0.70 | -0.004 | 0.026 | -0.011 | 0.028 | 0.072 | 0.025 |
| | 72000 | 0.025 | 14.5 | 2.4E+04 | -0.29 | 0.77 | 0.015 | 0.028 | -0.006 | 0.032 | 0.053 | 0.028 |
| | | 0.05 | 40.7 | 4.9E+04 | 0.25 | 0.69 | -0.076 | 0.055 | 0.032 | 0.047 | 0.017 | 0.046 |

Table IV reports the relationship between the synchronization error and the high frequency noise: the variability of $\mu_E$ is less evident with respect of the variability observed in Table III. Furthermore, the $\sigma_{\mu E}$ is stable with $T_S$, exception made for the case without radio latency.

Finally, the analysis made for the drift between the sender and the receiver clocks brings to the results reported in Table V. The imposed drift does not affect the value $\mu_E$ because it is fully corrected by the linear regression. The same behavior is observed for both cases with and without radio latencies: the only difference is in the magnitude of $\mu_E$ and $\sigma_{\mu E}$ achieved with the Bluetooth that have resulted always greater than ZigBee and WI-FI, thus following the trend observed in the previous analysis ("sub-section C").

Summarizing, the analysis carried out in this section brings to the following main considerations:

- The clock non-idealities, in terms of prescaler, drift, low frequency noise and high frequency noise, weakly influence the synchronization performance;

Table V Synchronization Error vs Drift

| | | | NO LATENCY | | BLUETOOTH | | WI-FI TCP | | WI-FI UDP | | ZIGBEE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | DRIFT [ppm] | $\mu_E$ [μs] | $\sigma_{\mu E}$ [ns] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] | $\mu_E$ [ms] | $\sigma_{\mu E}$ [ms] |
| $T_S$ [s] | 0.036 | 10 | 0.19 | 0.67 | 0.50 | 0.72 | -0.008 | 0.014 | -0.005 | 0.015 | 0.018 | 0.013 |
| | | 100 | 0.19 | 0.61 | 0.58 | 0.72 | -0.018 | 0.014 | 0.003 | 0.016 | 0.008 | 0.015 |
| | | 1000 | 0.19 | 0.57 | 0.60 | 0.65 | -0.018 | 0.016 | 0.005 | 0.015 | 0.005 | 0.015 |
| | 0.36 | 10 | 0.19 | 0.70 | 0.85 | 0.76 | 0.013 | 0.013 | -0.035 | 0.015 | 0.009 | 0.012 |
| | | 100 | 0.19 | 0.62 | 0.64 | 0.67 | -0.019 | 0.014 | -0.012 | 0.013 | -0.001 | 0.016 |
| | | 1000 | 0.19 | 0.69 | -0.06 | 0.64 | 0.005 | 0.014 | 0.002 | 0.016 | -0.017 | 0.012 |
| | 3.6 | 10 | 0.19 | 0.63 | 0.35 | 0.75 | 0.015 | 0.013 | -0.039 | 0.016 | -0.011 | 0.014 |
| | | 100 | 0.19 | 0.61 | -0.57 | 0.73 | -0.013 | 0.014 | 0.009 | 0.014 | -0.001 | 0.014 |
| | | 1000 | 0.19 | 0.65 | -0.21 | 0.75 | -0.022 | 0.016 | 0.002 | 0.014 | 0.003 | 0.013 |
| | 36 | 10 | 0.19 | 0.70 | 0.32 | 0.72 | 0.017 | 0.015 | -0.001 | 0.017 | -0.005 | 0.015 |
| | | 100 | 0.19 | 0.66 | -0.57 | 0.77 | 0.023 | 0.014 | 0.009 | 0.015 | 0.030 | 0.014 |
| | | 1000 | 0.19 | 0.58 | -0.28 | 0.73 | 0.019 | 0.016 | 0.008 | 0.015 | -0.004 | 0.013 |
| | 360 | 10 | 0.19 | 0.61 | 0.15 | 0.74 | -0.028 | 0.015 | 0.001 | 0.014 | 0.001 | 0.014 |
| | | 100 | 0.19 | 0.70 | 0.43 | 0.70 | -0.002 | 0.015 | 0.013 | 0.015 | 0.011 | 0.013 |
| | | 1000 | 0.19 | 0.65 | 0.94 | 0.68 | 0.001 | 0.015 | 0.014 | 0.015 | -0.013 | 0.015 |
| | 3600 | 10 | 0.19 | 0.64 | 1.3 | 0.73 | 0.010 | 0.015 | -0.002 | 0.015 | 0.011 | 0.014 |
| | | 100 | 0.19 | 0.57 | -0.03 | 0.69 | -0.021 | 0.013 | 0.027 | 0.016 | -0.002 | 0.015 |
| | | 1000 | 0.19 | 0.61 | 0.12 | 0.71 | -3.7E-04 | 0.015 | -0.011 | 0.014 | -0.007 | 0.015 |
| | 36000 | 10 | 0.19 | 0.68 | 0.62 | 0.65 | 0.002 | 0.013 | -0.014 | 0.016 | 0.019 | 0.015 |
| | | 100 | 0.19 | 0.59 | 0.38 | 0.71 | 0.008 | 0.014 | -0.007 | 0.015 | 0.008 | 0.013 |
| | | 1000 | 0.19 | 0.65 | -0.40 | 0.69 | -0.003 | 0.014 | 0.006 | 0.013 | 0.005 | 0.015 |
| | 72000 | 10 | 0.19 | 0.70 | -0.43 | 0.62 | -0.001 | 0.015 | 0.006 | 0.015 | 0.005 | 0.014 |
| | | 100 | 0.19 | 0.64 | -0.56 | 0.69 | 0.018 | 0.015 | -0.017 | 0.015 | -0.003 | 0.014 |
| | | 1000 | 0.19 | 0.72 | -0.76 | 0.71 | -0.008 | 0.014 | -0.012 | 0.015 | 0.003 | 0.014 |

- The radio latency is the main quantity of influence for the synchronization performance;
- Generally, TS values do not significantly worsen the synchronization error and the result repeatability.

## VI. RESULTS VALIDATION

A suitable measurement station has been designed to characterize the synchronization algorithms (see Figure 8). It is made up of the PC controller, the Tektronix TLA5202 Logic Analyzer, the Tektronix AFG3022B dual channel waveform generator, and two sensor nodes that can be connected through a either wired or wireless link.

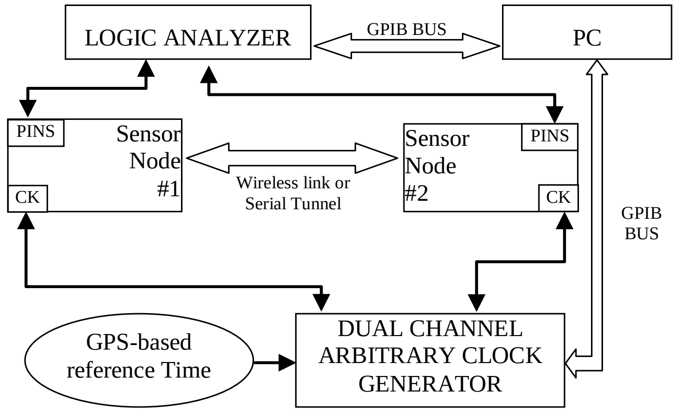


Figure 8 Block diagram of the measurement station for the characterization of the synchronization procedure.

The PC controller embedded in the logic analyzer but represented as an external unit for the sake of clarity, supervises all the instruments in the measurement setup.

The logic analyzer captures the time related to each happened event during the execution of experimental tests as better described below.

Both realized sensor nodes provide three specific output pins, whose signals were captured by the logic analyser, very useful for the debug and time analysis of the synchronization performance [11]. By this pins it is possible to measure the synchronization duration on each node (sender and receiver), to measure the synchronization error, and to estimate the times the sender node dispatches the first bit of the time-stamp packet and the receiver collects the first byte of the time-stamp packet. By this pin, it is possible to measure the packet transmission/reception duration. A suitable parameter was defined and analyzed during experimental tests, it is the clock phase delay (CPD) expressed as the mean time interval between the rising edges of the square wave transitions of the sender and the receiver node clocks after the synchronization procedure (see Figure 9). Further details on this parameter can be found in [10], [11]. Once such a parameter has been collected, it is possible to estimate $\mu_E$ and $\sigma_{\mu E}$ (having assumed the clock of the sender as reference).

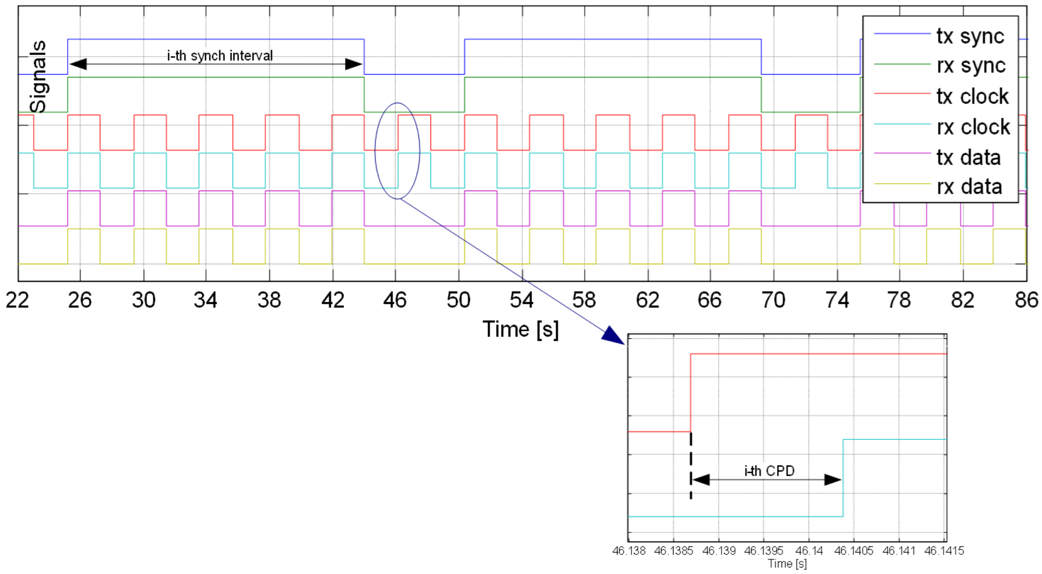


Figure 9 Captured signals for the evaluation of the algorithm performances (top) and selected figure of merit (bottom).

The dual channel waveform generator provides clocks to the two sensor nodes involved in the measurement setup during the tests performed in emulation environment. It is possible to highlight in Figure 8 the presence of a GPS-based controlled frequency standard (namely a Fluke 910RTM) that is able to deliver a precision frequency and pulse-per-second time reference to the waveform generator.

The whole measurement station has been placed inside the shielded and semi anechoic chamber of the University of Cassino and Southern Lazio, with the aim of avoiding any external electromagnetic interference.

### A. Experimental results in the case of emulated clocks and wired communication channel

First tests are aimed in considering the effect of clock non idealities in the presence of communication channel characterizes by negligible variable latency. To this aim the proposed experimental set-up considers an emulated scenario in which:

(a) the receiver node clock realizes three different drift levels which values are reported in Table VI, and with the high frequency noise and low frequency noise at the maximum values considered in Table I.

Table VI Analysis of the effect of the clock non ideality in the case of communication channel with negligible variable latency. The low frequency noise and the high frequency noise have been set at their maximum value of Table I.

| Imposed Drift | $\mu_E$ | $\sigma_{\mu E}$ |
|---|---|---|
| [%] | [μs] | [μs] |
| -0.625 | 72 | 11 |
| 0 | 57 | 10 |
| 0.625 | 82 | 12 |

(b) the communication channel is realized through a serial link at 38400 bps with negligible variable latencies and loss of packets.

More in detail, the sender clock has been fixed to 8.000 MHz while the receiver clock has been varied from 7.950 MHz (-0.625 %) to 8.050 MHz (0.625 %). This range is very wide if compared with the typical drift and accuracy of commercial clock that is typically bounded to 30 ppm, but it is very useful to highlight the synchronization performance versus the clocks drift.

The synchronization procedure has been carried out by sending 8 consecutive time-stamps every 2 s and then calculating the regression coefficients. A total time of about 16 s is required per synchronization. Each time-stamp packet is formed by 7 bytes, among these, 5 bytes describe the microcontroller timer evolution and 2 bytes represents the header and the checksum of the message. A total number of 200 consecutive tests have been executed for each receiver clock frequency.

Table VI describes the behavior of $\mu_E$ and $\sigma_{\mu E}$ versus the imposed drift. Results can be compared with those reported in Figure 5 when a Ts of 3.6 s is considered. Some considerations can be drawn:

i) the obtained values of $\mu_E$ are compliant with those reported in Figure 5 showing a mean value of about 70 us.

ii) As expected results in presence of positive and negative drift are very similar since, fixing the drift value, the algorithm is capable of compensate the drift sign.

iii)

Table VII. Analysis of the effect of the Radio Latency. Real clocks that exhibit drift, low frequency noise and high frequency noise are considered. A constant baud rate of 38400 bps is involved.

| Applied Maximum Radio Latency | $\mu_E$ | $\sigma_{\mu E}$ |
|---|---|---|
| [ms] | [ms] | [ms] |
| 5 | 0.12 | 0.10 |
| 16 | 0.13 | 0.29 |
| 32 | 0.16 | 0.64 |
| 50 | 0.67 | 0.90 |
| 64 | 0.8 | 1.3 |
| 128 | 2.3 | 2.5 |

iv) As expected in the case of absence of drift the mean value of $\mu_E$ reduces since only the effect of low frequency noise and high frequency noise acts on the synchronization performance.

**B. Experimental results in the case of real clocks and emulated communication channel**

Further tests aimed in considering the effect of real clock and the presence of communication channel characterizes by not negligible variable latencies are presented in the following. To this aim the proposed experimental set-up considers an emulated scenario in which:

(a) the sender and receiver nodes adopt real clocks;

(b) the communication channel is realized through a serial link at 38400 bps with six values of variable latencies.

As for the (a) point, the nominal frequency of these clocks was 8 MHz with accuracy of 30 ppm. A preliminary experimental characterization of these clocks has been made by evaluating their evolutions over a time of about a week. In particular, the following mean clock values ($\mu_{TC}$) and experimental standard deviations ($\sigma_{TC}$) have been estimated: $\mu_{TC}$ equal to 8.0002851 MHz and $\sigma_{TC}$ 0.2 Hz for the clock of the sender, $\mu_{TC}$ equal to 7.9993412 MHz and $\sigma_{TC}$ equal to 0.2 Hz for the clock of the receiver, respectively. These clock values bring to an absolute clock drift between the sender and the receiver of about 0.012percentage.

As far as the (b) point is concerned six variable latencies have been considered. These considered values well emulates the latencies expected when commercial ZigBee, Bluetooth, or WI-FI devices are employed.

The results related to this analysis are reported in Table VII and can be compared with those obtained in Figure7 when a Ts equal to 3.6 s is considered.

Some considerations can be drawn:

i) the obtained values of $\mu_E$ are compliant with those reported in Figure 7 for all the considered variable latency values;

ii) as expected as the variable latency increases the values of $\mu_E$ and $\sigma_{\mu E}$ increase.

### C. Experimental results in the case of real clocks and emulated communication channel

A test to analyze the effects of a real wireless communication channel and of real clocks has finally performed. In this case the same clocks of the previous case have been considered while the wired connection between the sensor nodes has been replaced by a wireless real ZigBee-based link [9]. As for the wireless module, it is based on the EasyBee board with the Atmel ATZB-24-A2 transceiver.

The sensor nodes have been placed at a distance of about 5 meters inside the shielded chamber and the temperature measured during the test varied in the range [20 °C, 22 °C].

The obtained results are reported in Table VIII and can be compared with those reported in Figure 7a).

Table VIII Experimental results on real nodes.

| $T_S$ | $\mu_E$ | $\sigma_{\mu E}$ |
|---|---|---|
| *[s]* | *[ms]* | *[ms]* |
| 0.082 | 0.8 | 0.9 |
| 4.096 | 6.2 | 3.1 |
| 20.972 | 11.2 | 7.5 |
| 104.858 | 52 | 46 |

Also in this case obtained results show values of $\mu_E$ of the same order of magnitude of the ones experienced in simulation environment (see Figure 7), and once again $\mu_E$ worsens as $T_S$ increases, thus confirming the trends experienced in simulation and emulation environments.

## VII. CONCLUSIONS

A detailed characterization of factors that influence performance of synchronization algorithms based on linear regression and operating on low costs wireless sensor nodes has been carried out in the paper. In particular, the paper has investigated in both simulated and real scenarios: (i) factors that influence the clock performance, (ii) factors that influence the microcontroller on which the synchronization is performed, and (iii) factors that influence the communication channel.

As for the (i) point the presence of drift, low and high frequency noise, has been analyzed in detail. With regards to the (ii) point factors as the adopted prescaler that influences the finite resolution of the timer, the variability of ISR time and the internal latency of the serial tunneling have been analyzed in detail. Finally as far as the (iii) point is concerned the presence of variable latency due to the typical low range wireless communication standards as BT, WiFi both with TCP-IP and UDP protocols, and ZigBee have been considered.

In addition, starting from the well-known timing model valid for scenarios in which the timing at MAC level is available, we have extended such a model to the case in which this feature is not present (as it happens in low cost networks), thus introducing all the above-mentioned quantity of influence.

The analysis carried out in simulation and emulation environments and on real nodes brings to the followings considerations: all the fixed latencies are well compensated by the regression algorithm; all the variable latencies, due to the microcontroller side latency, radio latencies, and the non-ideal behavior of the wireless channel, worsen the synchronization performance even if their mean values are well compensated by the regression algorithm and, in particular, the mean value of the sender-receiver clock error ($\mu_E$) and its variability ($\sigma_{\mu E}$) increases as the range of latency increases.

As for the non-idealities of clocks, the drift is well compensated by the regression while low frequency noise and high frequency noise affect the synchronization error (an increasing of $T_S$ brings to a performance decrease).

The analysis and results obtained in this paper could be particularly useful for developers of low cost WSN networks requiring time synchronization between nodes. The factors affecting synchronization performance of linear regression algorithms on low cost platforms have been analyzed in detail. Practical indications on both the components compensated by such kind of algorithms and the ones that make worse the performance have been provided.

## REFERENCES

[1] Akyildiz et al., "Wireless sensor networks: A survey," Comput. Netw., vol. 38, no. 4, pp. 393–422, Mar. 2002.

[2] N. Bulusu and S. Jha, Wireless Sensor Networks: A Systems Perspective. Norwood, MA: Artech House, 2005.

[3] Sundararaman et al., "Clock synchronization for wireless sensor networks: A survey," Ad-Hoc Netw., vol. 3, no. 3, pp. 281–323, Mar. 2005.

[4] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," Proc. 5th Operating Syst. Design and Implementation Symp. 2002, pp. 147–163.

[5] S. Ganeriwal, R. Kumar, M. B. Srivastava "Timing-sync Protocol for Sensor Networks" SenSys 2003, November 5-7, 2003, Los Angeles, California, USA.

[6] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems. New York: ACM Press, 2004, pp. 39–49.

[7] L. Ferrigno, V. Paciello, A. Pietrosanto, "Low-cost visual sensor node for BlueTooth-based measurement networks," IEEE Transactions on Instrumentation and Measurement, vol.55, no.2. pp. 521- 527, April 2006.

[8] L. Ferrigno, S. Marano, V. Paciello, A. Pietrosanto, “Balancing computational and transmission power consumption in wireless image sensor networks” Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2005. VECIMS 2005. Proceedings of the 2005 IEEE International Conference on, 2005.

[9] L. Ferrigno, V. Paciello, A. Pietrosanto, “Experimental Characterization of Synchronization Protocols for Instrument Wireless Interface,” IEEE Transactions on Instrumentation and Measurement, Vol. 60, Issue: 3, 2011.

[10] L. Ferrigno, V. Paciello, A. Pietrosanto, “A step forward the online minimization of the synchronization events in TPSN”, in Proceedings of IEEE International Instrumentation and Measurement Technology Conference 2012, Graz, Austria, pp. 2780-2784, May, 13-16 2012.

[11] D. Capriglione, D. Casinelli, E. D’Orazio, L. Ferrigno, “Time synchronization based on linear regression in low cost WSNs: A performance analysis”, in proceedings of the IEEE International Workshop on Measurements and Networking (M&N), 2013, Napoli, 7-8 Oct. 2013, pp. 25-30.

[12] G. Giorgi, C. Narduzzi, “Some notes on the Synchronization of Low-Power Wireless Nodes”, in proceedings of IEEE International Workshop on Measurements and Networking Proceedings (M&N), 2013, pp. 19-24.

[13] D. Fontanelli, D. Macii, S. Rinaldi, P. Ferrari, A. Flammini, “A Servo-Clock Model for Chains of Transparent Clocks Affected by Synchronization Period Jitter”, IEEE Transactions on Instrumentation and Measurement, Vol. 63, Issue. 5, May 2014, pp. 1085-1095.

[14] R. Carli, G. Giorgi, C. Narduzzi, “Comparative analysis of synchronization strategies in sensor network with misbehaving clocks”, in proceedings of IEEE International Instrumentation and Measurement Technology Conference 2012, Graz, Austria, pp. 2617-2622, May, 13-16 2012.

[15] SPI Block Guide V03.06, Original Release Date: 21 JAN 2000 Revised: 04 FEB 2003, Motorola, Inc.

[16] AN10216-01 I2C MANUAL, Jean-Marc Irazabal – I2C Technical Marketing Manager, Steve Blozis – I2C International Product Manager, Philips Semiconductors March 24, 2003.

[17] UM10204 I2C-bus specification and user manual Rev. 5 — 9 October 2012.

[18] IEEE Std 802.15.1™-2005 (Revision of IEEE Std 802.15.1-2002).

[19] IEEE Std 802.15.4™-2006 (Revision of IEEE Std 802.15.4-2003).

[20] IEEE Std 802.11™-2012 (Revision of IEEE Std 802.11-2007).

[21] N. Golmie, “Wireless Personal Area Networks,” IEEE P802.15-01/317r0, 2001.

[22] J. Eker, A. Cervin, and A. Horjel, “Distributed wireless control using Bluetooth,” in Proceedings of the IFAC Conference on New Technologies for Computer Control, Hong Kong, November 2001.

[23] N. J. Ploplys, P. A. Kawka, and A. G. Alleyne, “Closed-loop control over wireless networks,”IEEE Control Systems Magazine, vol. 24, no. 3, pp. 58–71, 2004. View at Publisher · View at Google Scholar.

[24] W. C. Craig, “ZigBee: Wireless Control That Simply Works,” ZigBee Alliance ZigBee Alliance, 2004, http://www.51zigbee.com/uploadfile/jh2.pdf.

[25] C. M. De Dominicis, P. Pivato, P. Ferrari, D. Macii, E. Sisinni, A. Flammini, “Time-stamping of IEEE 802.15.4a CSS Signals for Wireless Ranging and Time Synchronization”, IEEE Transactions on Instrumentation and Measurement, Vol. 62, Issue. 8, August 2013, pp. 2286-2296.

[26] P. Ferrari, G. Giorgi, C. Narduzzi, S. Rinaldi, M. Rizzi, “Time-stamp Validation Strategy for Wireless Sensor Networks Based on IEEE 802.15.4 CSS”, accepted for pubblication on IEEE Transactions on Instrumentation and Measurement.

[27] Definitions and terminology for synchronization networks, ITU-T Recommendation G.810, 1996.

[28] https://www.dropbox.com/sh/fvyvkj45m3wg9rx/AACrvf7-Qlhy7IVkiES95HNLa.

[29] J. Harrison, “Real Time: Some Notes on Microcontroller Interrupt Latency”, http://www.digikey.com/en/articles/techzone/2011/jul/real-time-some-notes-on-microcontroller-interrupt-latency