Evaluation of a Time Radio Signal as Wireless Sensor Synchronization Beacon

Dr. Meriel Huggard, Alessandro Vaccaro, Dr. Ciarán Mc Goldrick
School of Computer Science and Statistics
Trinity College Dublin, Dublin, Ireland
huggardm@scss.tcd.ie, vaccaroa@tcd.ie, ciaran.mcgoldrick@scss.tcd.ie

Abstract—Clock synchronization and universal time knowledge is a feature of which Wireless Sensor Networks are still in need. Reliable synchronization is necessary for applications involving distributed tasks, such as medium access control to sleep cycles for smart power management, precise sensor sampling, data sequencing and aggregation.

Signal processing of samples from multiple nodes is a practical application which benefit from a common time knowledge. In a scenario of Structural Health Monitoring is very important that all sensor sample the accelerometer data at the same time, to take a state snapshot of the monitored structure.

This article evaluates the MSF60 time broadcasting signal as a mean to synchronize the nodes of a WSN. A signal reception delay is observed and its impact of the system synchronization is evaluated through an experiment set. The experiment is based on an actual hardware and software implementation of the system. The article will conclude with an analysis of a possible application of the system to monitor low frequency events in sparse networks.

1 Introduction

Wireless sensor networks (WSN) mostly suffer from lack of time information therefore synchronization and knowledge about the exact sample acquisition time over several nodes is in many cases inaccurate. Timekeeping is on many WSN platforms performed through the micro-controller's hardware timers, which suffer from clock variations and are completely unrelated to time on other nodes of a network.

Hence, due to the lack of universal time information, the retrospective data samples correlation to time, during post processing, is challenging. Each data-set collected from the nodes of a WSN are hardly related to each other in time, in such conditions is impossible to have a real-time view of the evolution of a physical event over time and space.

As samples are correlated in time the sampling devices have to agree on a common clock source and converge their local clocks to it. Clock agreement is necessary for many distributed tasks, the aforementioned correlated sampling, time slotted media access, co-ordination of sleep cycles to prolong the battery life of sensor nodes. Time synchronization is therefore of paramount importance for a wireless sensor network. The new requirements posed by WSN deployments have raised interest in the research community into clock synchronization. As WSN nodes mostly run on batteries research has focused on simple, low bandwidth and low power synchronization techniques that roughly can be categorized in packet based synchronization and external source synchronization. Packet based synch diffuse the clock offset and time skew via network packets. Each node of the network evaluates its offset with a master clock by message passing. Several iteration might be necessary to achieve coordination and it can be maintained only with frequent re-synchronization. These techniques are effective on small well connected graphs, however do not scale well on big, sparse networks and to work properly the radio link must be available at all time. Are examples of this family protocols like RSB [1], TPSN [2] and FTSP [3]. The second family of synchronization approaches relies on an external (to the wireless node) reliable clock source. This may be a GPS receiver, the FM RDS signal [4], power lines AC electromagnetic pulses [5] and the national radio time signals such as DCF-77 [6]. These approaches have the non negligible advantage to allow synchronization without message passing, allowing their deployment on sparse networks or, excluding GPS based nodes, systems with very tight power requirements so that they need to reduce the wireless communications. The work

The hypothesis is that measurement data can be acquired simultaneously together with accurate universal time information on each wireless sensor, using the MSF60, a low frequency clock radio signal transmitted from the United Kingdom, as a sync beacon. This article aim to evaluate the quality of clock synchronization achieved on regular Arduino Duemilanove boards couples with cheap MSF60 receivers. We have run three set of experiments to evaluate the MSF60 signal precision, a second set to measure drift of two Atmega328 micro-controllers clocked at 16 MHz and a third set where the MSF60 receiver fed the Arduino Duemilanove boards as a clock sync source.

- The precision of the MSF60 signal is evaluated in two phases: first in subsection 4.1 was measured the actual duration of a second, then in subsection 4.1.1 were measured the timing differences between two independent MSF60 receivers.
- The clock drift of two non synchronized Arduino was measured over several rounds for up to half day, showing a growing offset up to 180ms between the two boards, see section 5.
- The Arduino clock is then corrected with a MSF60 time receiver and is shown how the errors measured in the previous experiments actually combine together, the result and discussion is in section 6.

In the next session is given an account on related work in the area followed by a description of the MSF60 universal time signal. Afterwards will follow three sections each describing in detail the mentioned experiments on said time sources. A conclusion will summarize the results of the experiments.

2 Related Work

The research described in [6] analyses the geographical availability of the broadcasted WWVB and DCF-77 signal and the power requirements needed to make the receiver module work. The radio signals are two time diffusion transmissions operated at low frequency in the USA and Germany respectively. They measure an inter-node synchronization skew variable between 1 mS and 11 mS. Their conclusion is that this technique lacks from good quality hardware which is not able to provide microsecond precision. However is still a good clock source to synchronize sparse networks to allow time slotted medium access for nodes with very tight power requirements.

Another work in the area [4] proposes to use the widely available RDS message broadcasted by most FM radio stations as a common time reference. Their main point is to avoid the poor signal reception of DCF-77 or similar technologies in indoor WSN deployments. The experiment was run on purposely designed hardware and a calibration algorithm has been implemented which can keep the offset between two nodes under 1,5 mS with an average of 0,4 mS. The calibration algorithm proposed could be of inspiration for a similar approach on Universal Time Signal Receiver based systems, given that the error feature a similar configuration.

Central time reference based synchronization solutions usually performs poorly compared to packet based synchronization protocols in terms of mere offset reduction. However according to [4] the trade off is on power consumption, converging delay, network size scalability and link availability. RBS [1], FTSP [3], TPSN (>20 mS precision) [2] achieve their accuracy with many repeated packet exchange or time flooding. These come at a power cost and they greatly limit the bandwidth available to applications for which the WSN is deployed. One of the main advantage of using a Universal Time Signal Receiver based synchronization system is the availability of global time, while even the RDS based solution gives only information about the relative offset between a set of nodes. The universal time information, would ease the correlation of signals from multiple sources without introducing further processing delays to infer event sequence out of relative offsets. Also the concept of universal time is useful to sparse networks that are then able to timestamp their tasks without to be physically linked to the other nodes of the network. Another work in the area uses constructive interference to facilitate network flooding [7] with the side effect of obtaining implicit time synchronization, reducing the offset to $0.5\mu S$.

3 Radio Time Signal Protocols

Global time keeping is necessary to synchronize the operations of several events. For this reason most countries have created a national institution to build and maintain a radio signal transmitter to broadcast time and other related information. In Europe the most common are the German DCF-77 signal and the British MSF-60 other European nations have their own transmitters. Other continents have some equivalent technology and a quick look at the technical documentation will show that most of them consist of an analogical signal that span for the length of a minute, transmitting a bit per second. These signals are therefore easy to decode also with simple hardware and software routines. Indeed time signal receivers are implemented in wall clocks as well as time keeping server.

The MSF-60 signal is transmitted from Anthorn Radio Station in Cumbria on the 60 kHz frequency. It consists of a one minute time frame modulated similarly to the DCF-77 protocol, each second of signal carries one bit. At the beginning of each second the carrier is switched off for 100 mS to encode a logical 0, for 200 mS for a logical 1. The switch to a new minute is signaled by a carrier drop of 500 mS. The relevant time information, as well as the parity and control bits, are BCD encoded in bits 17 to 59.

Fig. 1. This map roughly represents the area within which the MSF60 (orange) and the DFC77 (red) can be clearly received. This are the values reported by the documentation provided by the organizations responsible for the maintenance of the time signal broadcasts. However by our experience, using a cheap receiver, in Ireland the DCF77 signal is received with a lot of noise. For this reason our experiments are based on the MSF60 signal, which due to the geographical proximity of the transmitter, was received without any noise by our cheap receivers.

The DCF-77 protocol describes an analogical signal broadcasted from Mainflingen near Frankfurt, Germany. The signal is AM modulated on the 77,5 kHz frequency. It has a 59 seconds frame, one bit transmitted each second. The carrier amplitude is dropped of about 25% at the beginning of each second, a drop duration of 100 mS represents a logical 0 while a drop of 200 mS is a logical 1. The calendar day, day of the week, month, year, hour and minute information is encoded in BCD (binary coded decimals) in bits 20 to 59. Frame bits 28, 35 and 59 are three parity bits to check the correctness of the received frame.

Although the two protocols share the same BCD encoding they are not compatible and is necessary to write a specific decoder for each signal. However as demonstrated in this paper, it is not necessary to completely decode the bit stream to perform a clock synchronization. Indeed it is sufficient to identify the new minute pulse to have fairly stable clock reference. The experiment described in this paper was run in Ireland so the choice for the MSF-60 signal, being stronger than DCF-77 due to geographical proximity to the transmission site.

3.1 Geographical availability

The MSF-60 signal can be received at a distance of 1000 Km from Anthorn with a field strength of at least 100 μV/m. The radius is extended enough to cover the United Kingdom, Ireland, Denmark, Belgium, Nederland, south west Norway and northern parts of France and Germany. In figure 1 is sketched map of Europe with the estimated signal availability radius. A few times during the year the signal is switched off for maintenance [8], but the British Physics Laboratory provides notices for the scheduled temporary shutdowns.

The DCF77 radio signal, can be received from a distance of 2000 Km, it covers most of Europe and within 500Km the signal is received with a field strength of 1 μV/m [9]. This signal can

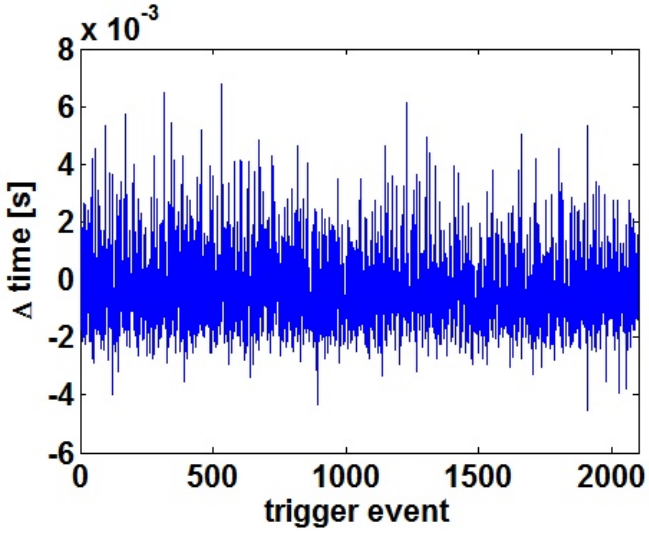

Fig. 2. Exists a variability in the second duration in the MSF60 signal. The plot shows the variability, in milliseconds, of the duration of each second. The error mostly fluctuates between ±2ms and does not seems to follow a predictable pattern.

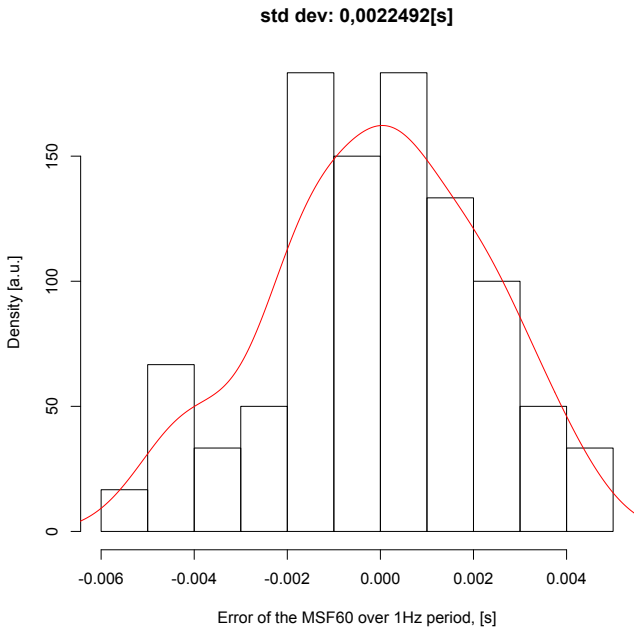

Fig. 3. This plot show the error on the duration of each second on the MSF60 signal, the vast majority of the samples lie ±2ms, the error although substantial, averages out on the long term.

be used effectively, as stable and reliable time source at over 1600km from the transmitter [10].

As is common for all long wave signals the range and quality of reception varies from night and day, the signal strength is higher at night time and in general in good weather conditions, however light rain has negligible effect on the long radio waves used by this technology.

4 MSF60 TIME SIGNAL PRECISION

Some of the main limitations of packet based synchronization techniques are network size scalability, data traffic overhead and power consumption. By adopting a central time reference is possible to dramatically reduce the number of packages transmitted to synchronize the nodes of a WSN. Consequently reducing the CPU time dedicated to process the protocol and maintaining idle the radio modem for longer periods, therefore power consumption is expected to decrease. However this should not happen at the expense of time accuracy.

Multiple experiments have been executed on different settings and configurations, to evaluate the suitability of Universal Time Signal Receiver in general, and of MSF-60 in particular, as providers of a common time reference. These experiments are described in the following sections.

4.1 MSF60 Signal Precision

The experiment was split in two parts dealing mainly with different methods of sampling the time samples directly off a couple of MSF-60 receivers and through a couple of Arduino Duemilanove. It is important to know the quality of the clock source that will be used to build a synchronization system. Therefore in this chapter are described the details of the two experiments executed to analyze the stability and predictability of the MSF60 signal.

4.1.1 Single MSF60 receiver setup

A single MSF60 Receiver, sampled at the output signal over a few hours to calculate the average length of a second and a minute.

The first experiment was run using one MSF-60 receiver and a logic analyzer connected to the output pin of the MSF-60 receiver itself. The output signal was sampled with a frequency of 25KHz for 3600 seconds. The collected data was then processed to calculate the average length of each second and minutes in the form of output of the receiver board. This test was particularly useful to evaluate the precision and repeatability of the MSF-60 time signal.

The experiment results showed a variability and unpredictability of the length of a second (1000ms) in the range of ±2ms for the vast majority of the samples as is possible to appreciate in figure 2. The time signal encodes the beginning of each second as a raising edge, but the frequency of this event is not 1Hz as expected although it is very close. The error is probably introduced by noise due to the cheap radio receiver used. In figure 3 is plotted the time length distribution of the sampled signal, showing that the duration of the signal follows almost a normal distribution.

4.1.2 Two MSF60 receivers setup

The time signal was sampled in parallel on the two devices for 3600 seconds. The objective of the experiment was to evaluate the average (and standard deviation) output line state transition skew between the two receiving devices. From the data collected has emerged that the output signal from the two MSF-60 receivers has a median skew of 0.00172 s with a standard deviation of 0.00229 s. Over the sampling period the highest delay registered was of 0.02536 s and the minimum 0.00004 s. The plot in figure 4 represent the delay between two receivers and it visually shows that the it is not predictable and does not seems to follow any repetitive pattern. Probably the imperfect correlation is due to imperfections in the electronics of the receiver boards, as well as not perfectly tuned crystals. However a set of wireless sensor nodes equipped with this receivers would still be able to synchronize accurately enough

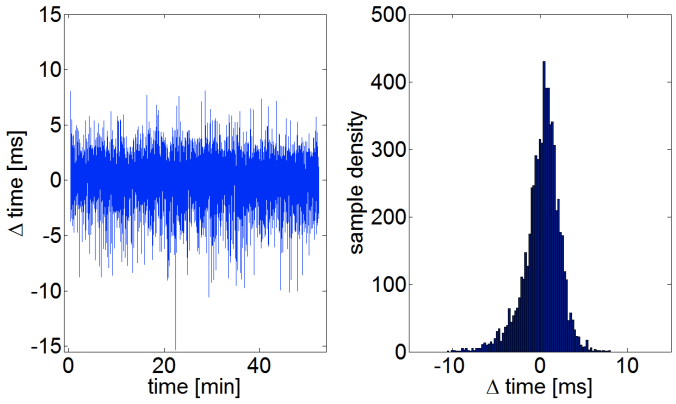

Fig. 4. Reception delay of the MSF60 signal from two different receivers. The plot show the difference between the second pulse as received by the radio. It show how the signal output from the two receivers has a random and unpredictable offset. However it is stable inside acceptable margins and if considering a 1 minute period the error averages out.

for most low frequency operations and also to schedule the sleep cycles of the nodes.

4.2 Time-stamping Procedure

The MSF60 bit stream is a composed of 59 bit and it is BCD encoded. It contains information about the current hour, day of the year, day of the week, month and year. There are some special bits to report a following leap second and if Summer Time is in effect. All these information are not relevant to the objective of providing a WSN with a clock source and to demonstrate that it is indeed a reliable source.

The experiment is run on two Arduino Duemilanove, both keep a local clock that every 1000 ms raises an interrupt served by the routine sketched in algorithm 1. This routine counts how many seconds have elapsed since the board was powered on, it also counts the number of minutes and execute a clock synchronization task every ten minutes. The main event loop running on each Arduino is a finite state machine sketched in algorithm 2. Every ten minutes the system clock routine will move the state machine to a state called MINSIG which will start waiting for a new minute pulse coming from the MSF60 receiver.

The whole code is very compact and simple, to time stamp events is provided a simple C macro that produce a time stamp reading the variables tm_sec, tm_min and the current system timer value.

5 Clock Drift between two Arduino

To better evaluate the benefit brought by a global clock synchronization in terms of clock drift reduction, we have sampled the clocks of two Arduino Duemilanove. The sampling was executed during a period of over 12 hours by mean of a portable logic analyzer. Both test boards were placed next to each other, to minimize the effect of environmental temperature changes to the individual clocks as the onboard clocks are not temperature compensated. On the Arduino Duemilanove is installed an ATmega328, running at 16MHz, on which are available two 8 bit timers and one 16 bit timer. For this phase of the experiment was used the 16 bit timer operating in CTC mode the pre-scaler register set to 256 to ignite an interrupt every second. The code executed on the boards was a simple

Algorithm 1 This interrupt routine is called with a frequency of 1Hz and handles the system timer. It keeps track of the elapsed seconds and moves the state machine into synch state every 10 minutes.

```
function TIMER_HANDLER ▷ interrupt every second
    tm_sec ← tm_sec + 1
    if (tm_sec mod 60) = 0 then
        tm_min ← tm_min + 1
    end if
    if (tm_min mod 10) = 0 then    ▷ synch every 10 min
        state ← MINSIG
    end if
end function
```

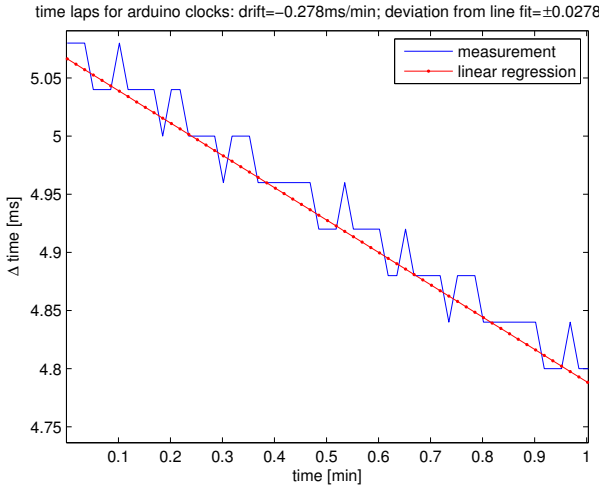

Fig. 5. A zoomed in plot of a 12 hour sampling of two free running Arduinos clocks. It is visible a small deviation of ±0.03ms from the linear regression and a drift of 18ms/h.

ISR routine that toggled one of the GPIO and increased an elapsed second counter used also to toggle another GPIO at the beginning of each new minute. This part is equal for both boards. To precisely start both timers at the same time one Arduino was programmed as a Master that start its clock and send a start command to the second Arduino programmed as Slave. The start command is a simple impulse on one of the available IO pins. An external interrupt ISR on the slave board detects the pin state change and start its own timer. The initial clock offset is negligible due to this setup. The offset grew of about 18 ms/hour. It is possible to notice the drift on one of the two ATmega328 grows faster and consistently from the other. The clock offset diverge linearly from the original value, with an offset of over 180 ms in the last hour of the experiment, in figure 5 is visible the offset density which is equally distributed due to the particular environmental conditions in which the experiment was run. In the next section is shown how the MSF60 receiver effect the clock offset stability.

6 Clock Correction with MSF60 Signal

In the previous section was shown how the clocks of two Arduino drift apart over time to grow the clock offset up to 180ms. In this section we show how it is possible to control the offset and keep it within a certain threshold. The experiment

setup is composed of two Arduino whose clock is regularly corrected against the signal of an MSF60. The signal is fed into the Arduino one via one of the digital GPIO. The two test boards were placed indoor, in a brick building, without the steel cage of armed concrete that stops the radio waves [6]. Each board had its own receiver and was completely independent from the other. In this experiment each board is programmed to emit a pulse of the duration of 1000 ms every second and a pulse of 60000 ms every minute. These pulses have been sampled with a logic analyzer and the offset evaluated a posteriori.

6.1 Synchronization Code

The clock is synched with the MSF60 at power on and then every 10 minutes. During the synchronization window the Arduino waits for a new minute pulse and then reset its own clock to match the rising edge of the minute pulse. The pseudocode in 2 describes more accurately the state machine that handles the timer reset and that looks for the new minute pulse of the MSF60 signal. The finite state machine is controlled externally by two interrupt service routines, described in code 3 and 1. The system timer routine (pseudocode 1) is invoked every 1000 ms, increases the second counter and every 10 minutes moves the state machine into the MINSIG state (minute signal). The MINSIG state enables the external interrupt on the falling edge of the pin state change and moves the FSM into WAIT. From now on the FSM is controlled by the external interrupt routine (pseudocode 3) that is called when a falling edge is detected on the MSF60 signal stream, starts timer B and moves the state machine into FALL_EDGE state which sets the external interrupt to react on rising edges. The code so only detects a logic 0 of about 500 ms between a falling edge and a rising edge. This operation is justified to catch the bit with the switch over pulse encoded in the first second of each new frame of the MSF60 stream, and is encoded switching off the carrier amplitude for 500 ms at the switch over to the next frame which also mark the beginning of the next minute.

6.2 Offset Reduction

The length of the synchronization window can not be predicted, in the best case lasts at most 500ms, which is the duration of the new minute pulse, however in case of bad radio reception or jammed signal the Arduino might have to wait for a few minutes. The plotted data shows that synchronizing the Arduino timer with the MSF60 signal the offset is reduced and kept within the precision boundaries of the MSF60 receiver. In figure 6 is plotted the error density of the resulting experiment. The offset is contained within -10 ms and 5 ms, enough precision to timestamp events that have a frequency of less than 100Hz. In figure 7 is plotted the offset over the duration of the experiment, the spikes are due to hardware issues with one of the MSF60 receivers that is less stable in receiving the radio signal. To prove this statement was run a test where both Arduino boards were fed with the same MSF60 output, the outcome is visible in figure 8. This test excludes that the sudden offset growth were due to either the code or the Arduino boards, also the big offset skew monitored in figure 7 are not consistent with the measurements results visible in figure 4.

Algorithm 2 The main event loop. It is a state machine to control the two timers and the MSF60 radio receiver.

```
while true do
    if state = MINSIG then                          ▷ Initial state
        ExtInt ← fall_edge
        state ← WAIT
    end if
    if state = E_FALL then                          ▷ Pulse start
        ExtInt ← rise_edge
        state ← WAIT
    end if
    if state = E_RISE then                          ▷ Pulse end
        if timer_b_read() = 500 then                ▷ Check length
            state ← RESET
        else
            state ← MINSIG
        end if
        timer_b_reset()
    end if
    if state = RESET then
        timer_a_reset()
        ExtInt ← Disable
        state ← WAIT
    end if
    if state = WAIT then
        wait_state_change()
    end if
end while
```

Algorithm 3 The interrupt routine that looks for the new minute pulse from the MSF60 receiver. It measures the pulse length and advances the state machine in the main loop.

```
function ISR_EXT_INT
    if ExtInt = fall_edge then
        timer_b_start()
        state ← E_FALL
    else
        timer_b_stop()
        state ← E_RISE
    end if
end function
```

7 Results

We present the results of a set of experiment to evaluate the precision of radio signal time references in general and of MSF60 in particular. An experiment is executed to measure the stability of the time information provided by the signal, which shows an average oscillation of 0ms±2.2ms on the 1Hz period. It is also measured, over a period of 3600 seconds, a skew in the output data of two independent MSF60 receivers of 1.72ms±2.29ms.

In the second set of experiments was evaluated the clock drift of two Arduino Duemilanove boards. The measurement lasted for over 12 hours and the Arduinos had no mean to communicate, apart from the initial synchronized timer start, during which was measured a drift of 18ms/h±30μs deviation from the linear regression.

The last test used the radio signal time reference to reduce the Arduinos clock's drift. Each Arduino board is connected to the output of a radio time signal receiver and at regular intervals the internal clock of each board is adjusted with the new minute pulse. With this setting is measured that the clock offset between the two Arduino boards is consistently kept within 3.2ms±3.5ms.

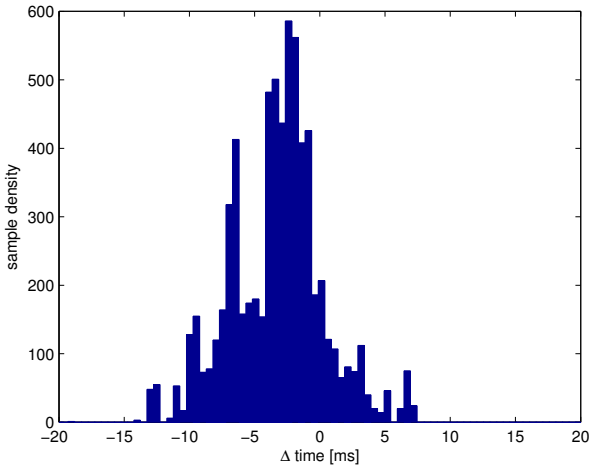

Fig. 6. The MSF60 corrected clock offset density plot of the samples collected from two independent Atmega328 mcu. The offset is contained within 5 and -8 ms.

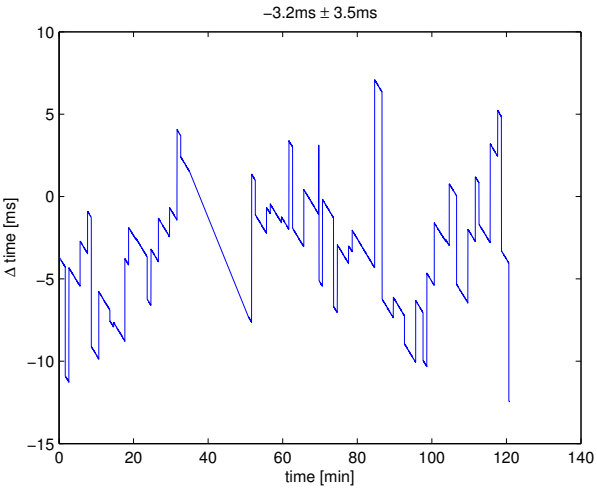

Fig. 7. The plot of two hours of samples of the clock of two Arduinos show that the radio time signal reference synchronization keeps the offset at 3.2ms±3.5ms from the linear regression.

8 Conclusion

It is presented a technique to synchronize outdoor wireless sensor networks by mean of a global radio signal time reference, in particular the British broadcasted MSF60 signal. It is evaluated the precision and consistency of the time signal showing fluctuations within 0ms±2.2ms on the 1Hz period. It is also measured the clock drift of two Arduino over a period of 12 hours. On the last test the Arduinos clock were regularly synchronized to the radio time signal by a software routine, measuring that the offset is kept within 3.2ms±3.5ms. Each experiment was run indoor for a period of 12 hours, in the future, it is anticipated to test long term deviations. The results of herein presented experiments demonstrate that time synchronization better than 10ms can be achieved using radio broadcasted signal time references and cheap radio receivers.

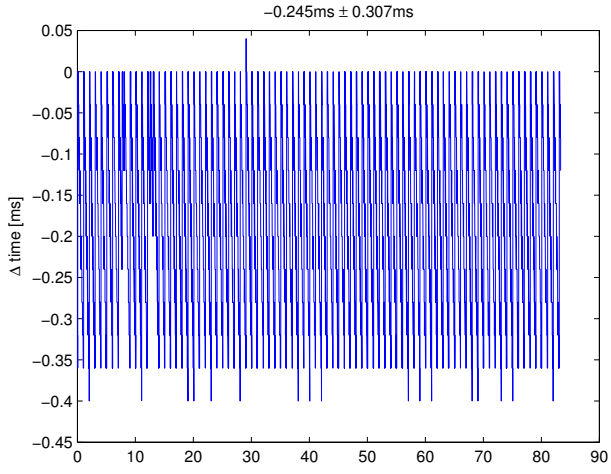

Fig. 8. If the system clocks are fed with the same MSF60 output their clock offset is contained within $0,5$ ms. This test was run to exclude as cause of the occasional offset spikes to either the code or the Arduino boards.

References

[1] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 147–163, 2002.

[2] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, ser. SenSys '03. New York, NY, USA: ACM, 2003, pp. 138–149. [Online]. Available: http://doi.acm.org/10.1145/958491.958508

[3] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, 2004, pp. 39–49.

[4] L. Li, G. Xing, L. Sun, W. Huangfu, R. Zhou, and H. Zhu, "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*. ACM, 2011, pp. 169–182.

[5] A. Rowe, V. Gupta, and R. R. Rajkumar, "Low-power clock synchronization using electromagnetic energy radiating from ac power lines," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '09. New York, NY, USA: ACM, 2009, pp. 211–224. [Online]. Available: http://doi.acm.org/10.1145/1644038.1644060

[6] Y. Chen, Q. Wang, M. Chang, and A. Terzis, "Ultra-low power time synchronization using passive radio receivers," in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*. IEEE, 2011, pp. 235–245.

[7] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, April 2011, pp. 73–84.

[8] (2010, 04) National physical laboratory: Msf radio time signal. [Online]. Available: http://www.npl.co.uk/science-technology/time-frequency/products-and-services/time/msf-radio-time-signal

[9] (2010, 3) Physikalisch-technische bundesanstalt (ptb): Reach of dcf77. [Online]. Available: http://www.ptb.de/cms/en/fachabteilungen/abt4/fb-44/ag-442/dissemination-of-legal-time/dcf77/reach-of-dcf77.html

[10] P. Dolea, P. Dascal, T. Palade, and O. Cristea, "Aspects regarding the use of lf radio transmitters for time dissemination," in *Electronics and Telecommunications (ISETC), 2014 11th International Symposium on*, Nov 2014, pp. 1–4.