# “Real” Signal Processing with Wireless Sensor Networks

György Orosz, László Sujbert, Gábor Péceli
Budapest University of Technology and Economics, Department of Measurement and Information Systems
Magyar tudósok krt. 2., Budapest, H-1521, Hungary
Phone: +36 1 463-2057, fax: +36 1 463-4112, E-mail: orosz@mit.bme.hu

**Abstract – This paper introduces some practical issues of the deployment of wireless sensor networks (WSNs) in closed loop digital signal processing (DSP) and control systems that require relatively fast, continuous data flow and real-time operation of the data collecting network, as in the case of traditional systems where sensors are attached by wire.**

**The difficulties of the design of the wireless control and signal processing systems are introduced through a particular application that is an active noise control system. Main topics that are presented are the scheduling of tasks, synchronization and distributed signal processing.**

## I. Introduction

This paper deals with the application of wireless sensor networks (WNSs) in relatively high speed feedback signal processing systems. The design of these systems requires different approach than typical applications of wireless sensor networks (e.g. health or environmental monitoring [2]), where the observed signals change relatively slowly (e.g. temperature, humidity), and the interactions that have to be performed are not time critical. Computations and evaluation of results can also be performed offline, or at least with soft time limit. However, in the traditional digital signal processing (DSP) systems—which can be called “real” signal processing systems—signal processing algorithms run real time, and expect the data to be processed from the data sources continuously with the sampling frequency of at least some kilohertz. In the case of feedback systems the situation is even worst, since the interactions are carried out according to the input signals. If inputs are provided by the WSN, faults in the network can cause even the instability of the whole system. In these DSP systems the widespread methods for the general applications of the sensor networks can fail, and they are no more suitable for their original purpose [1].

Despite of the difficulties, the research of the field of wireless signal processing is promising because of the obvious advantages of the utilization of the WSN’s [2]. These attractive features of WSNs are e.g. the easy installation and maintenance of sensors, the flexibility of the arrangement, and the radio communication that eliminates the costs of cabling.

The difficulties of the design of the wireless control and signal processing systems are introduced through a particular application. In this case study some variants of an active noise control (ANC) system [3][4] are presented as typical instances of wireless signal processing systems.

The structure of the paper is the following. In Section II. the hardware and software configuration of the wireless signal processing system is described. The common properties of ANC systems are presented in Section III. The difficulties in connection with the scheduling of tasks are discussed in Section IV. The importance of the synchronization and typical solutions for the synchronization are showed in Section V. Section VI deals with the data transmission and signal processing properties of the realized systems putting emphasis on data reduction.

## II. System Description

### *A. Hardware Configuration*

The block diagram of the designed active noise control system that utilizes wireless sensors can be seen in Fig. 1. The system basically consists of two units. The main parts of signal processing algorithms are implemented on a DSP evaluation board of type ADSP-21364 EZ-KIT LITE [5], which includes an ADSP-21364 (SHARC) processor. The DSP is a 32 bit floating point one with a maximal clock frequency of 330 MHz and dual arithmetic units. The DSP is connected to an AD1835 codec that has two analog input and eight analog output channels, through which signals can be fed to the loudspeakers. Such number of output channels ensures the possibility of realization of extensive systems. The analog inputs can be used for collecting reference signals that is required for most ANC systems. Reference signals can be collected also by motes. The acoustic signal is sensed by the elements of WSN, which is built up of Berkeley micaz motes [6]. These motes are intelligent sensors that consist of an ATmega128 eight bit microcontroller with a clock frequency of 7.3728 MHz, a CC2420 2.4 GHz ZigBee compatible radio transceiver and an MTS310 sensor board. The data transfer rate of the transceiver IC is 250 kilobit per second (kbps) including a preamble section, a header and a footer that are handled by hardware. The sensor board includes also a microphone with variable gain amplifier, the output signal of which is converted by a 10 bit analog to digital converter (ADC) of the microcontroller.

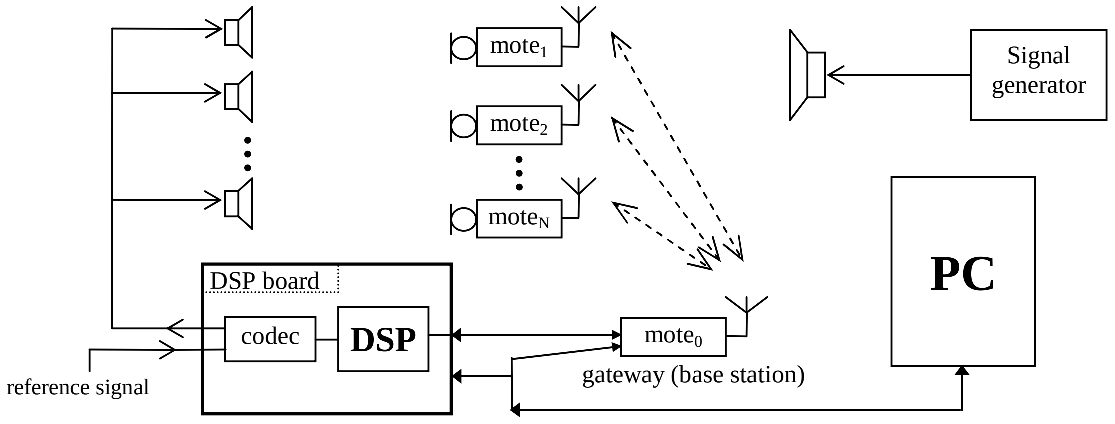


**Fig. 1**. Block diagram of the system

Most of motes (mote$_1$…mote$_N$ in Fig. 1) are responsible for noise sensing. They transmit data towards the DSP. Data from the wireless network are forwarded to the DSP by a gateway mote (mote$_0$ in Fig. 1). The DSP and the gateway mote are connected via asynchronous serial port. The transmission rate of communication between the two units is 115.2 kbps. The programming of motes is carried out by an interface board of type MIB510 [6]. It serves as a power supply and RS232 line driver for gateway mote, as well.

In order to ensure the reliable and real time operation of the system—that is one of the main tasks in the control system—a Time Division Multiplexing (TDM) network operation was utilized: the network operation is divided into periods, in which each mote transmits data to the gateway in succession in predetermined order and time instants.

The PC basically serves as developing and debugging tool for both platforms. Additionally, it is suitable for logging and visualization of data sent by gateway or DSP over serial port.

An independent loudspeaker driven by a signal generator can be utilized to generate external sound. It can be used as an artificial noise source while testing the ANC system, or as general examination or excitation signal for test purpose.

### *B. Software Components*

The software for the DSP was developed with the VisualDSP++ software developing environment provided for Analog Devices DSPs. The main programming language is C but assembly subroutines are also used.

For software development for motes the TinyOS embedded operating system (OS) was used, but modifications should have been carried out in order to increase the code efficiency. Typical difficulties emerge in time critical sections, where accurate timing of tasks is inevitable. Such tasks are timing of the sampling the microphone's signal, and accurate detection of arrival time of radio messages, which is important task in synchronization. The OS provides convenient interface for handling the hardware elements e.g. radio, AD converter.

The programming language of the TinyOS is nesC [7] that is a component based language. The programs are built up of components (or modules) that are connected to each other via interfaces. These components are provided with the OS, but components can also be created by the user of the OS. Each component is responsible for a certain task e.g. handling of the radio or microphone, and the user code is also included in an independent component. The interfaces include *events* and *commands*. Via events one component can notify another one of certain events (e.g. the arrival of a radio message). Via commands one component can use the services of another component (e.g. sending a message).

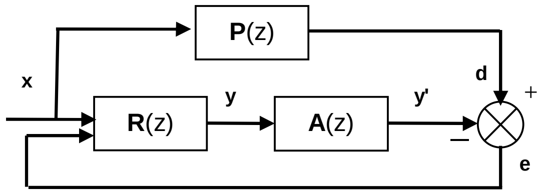
The scheduling entities of the OS are the following in descending order of priority (beginning with the highest priority):

1. HW interrupts
2. Events and commands
3. Tasks

The HW interrupts can interrupt each entity. Events and commands can interrupt each other and the tasks. Tasks can not interrupt any scheduling entity.

## III. Introduction Of Active Noise Control

The general linearized model of ANC systems can be seen in Fig. 2.



**d**: primary noise
**y**: secondary noise
**e**: error signal: $\mathbf{e} = \mathbf{d} - \mathbf{y}\cdot\mathbf{A}(z) = \mathbf{d} - \mathbf{y'}$
**A**($z$): transfer function of secondary path
**P**($z$): transfer function of primary path
**R**($z$): active noise control algorithm
**x**: reference signal

**Fig. 2.** General structure of ANC systems

The purpose of ANC systems [3] is to suppress mainly low frequency acoustic noises by means of destructive interference. The operation can be summarized as follows: according to the reference and the error signals (**x** and **e** respectively), the ANC algorithm **R**($z$) produces such secondary noise **y** which minimizes the power of the error signal **e**. **R**($z$) is implemented mainly on DSP because of its computational complexity. **y** is radiated by loudspeakers and arrives to the microphones through the so called secondary acoustic path that is described by an **A**($z$) matrix which consists of the transfer functions between each output and each input of the signal processing algorithm **R**($z$). **A**($z$) can be treated as a linear system. The error signal **e** (i.e. remaining noise) is the result of the interference of the primary noise **d** and secondary noise **y'**, and it is sensed by microphones.

In the ANC algorithm a kind of inverse model of the secondary path **A**($z$) is applied which is denoted by **W**($z$). For ensuring the stability of the system, **W**($z$) is often chosen as follows [8][9]:

$$\mathbf{W}(z) = \mathbf{A}(z)^{\#} \tag{1}$$

where $^{\#}$ denotes the pseudo- (or Moore-Penrose) inverse. The secondary path **A**($z$) should be identified in advance. The stability criterion of the system is:

$$-\pi/2 < \mathrm{arc}(\lambda_l) < \pi/2 \tag{2}$$

$$\lambda_l = \lambda_l(\mathbf{A}(z)\mathbf{W}(z));\quad l=1\ldots L \tag{3}$$

where $L$ is the number of inputs. Eq. (2) and (3) means that all eigenvalues $\lambda_l$ of the term **A**($z$)**W**($z$) must have positive real part for each frequencies, where noise is present. For single channel case (only one microphone and one loudspeaker are used) it means, that the phase delay of **A**($z$) must be known at least with the accuracy of 90$^o$. **A**($z$) involves the transfer function of the entire signal path between the output and input of the noise control algorithm (i.e. ADC, DAC, analog signal conditioner circuits, transfer functions of microphones and loudspeakers, and the transfer function of WSN's data routing is also included). The main goal in ANC systems (and in our case also in the WSN) to ensure the permanency of **A**($z$) on each frequency, because so can **W**($z$) be determined unambiguously and utilized during the entire operation time of the system.

ANC systems can be regarded as model adaptive control systems. The reference signal **x** comes from the noise source and is utilized by the ANC algorithm. **x** passes also through the secondary path **P**($z$) and comes to the microphones. The aim is to make the output of the acoustic system (**y'**) to be equal to the output **d** of the reference system **P**($z$), so the equivalence of **R**($z$)**A**($z$) and **P**($z$) has to be ensured by the controller **R**($z$) as far as possible.

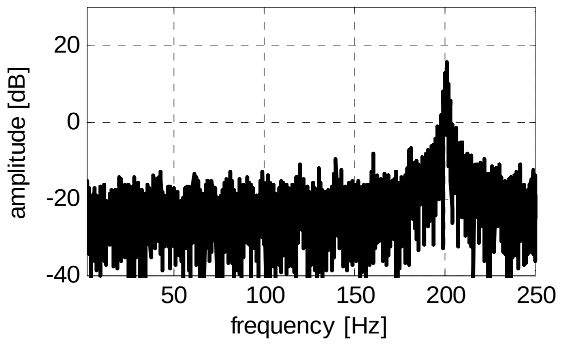
## IV. Effects Of Task's Piority On Signal Sensing

In this section the importance of the priority of the tasks in the systems is introduced, putting special emphasize on the timing of the sampling that plays important role, since the observed acoustic signal changes relatively fast.

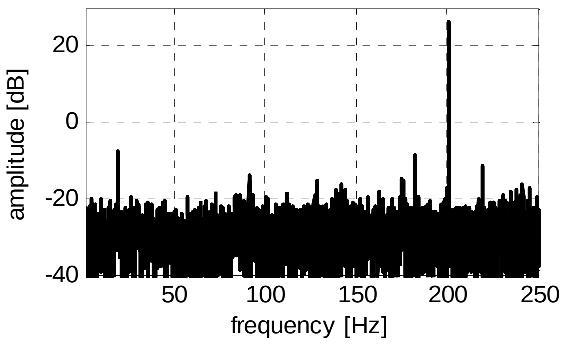
The operating system (OS) provides more possible ways for the timing of certain processes. These methods use the hardware timers of the microcontroller in different ways.

If a software timer component is utilized that uses the same hardware timer for serving more tasks through different interfaces, this would result in the degradation of the accuracy of the timing. This is because at the interrupt of the hardware timer the software timer component has to choose on which interface it has to signal an event. The time interval $T_{\mathrm{delay}}$ between the real hardware interrupt and the generation of the event on a timer interface is not deterministic, since the components connected to the timer component may request timing event at arbitrary time instants, so the timing requests of different components disturb each other. In Fig. 3.a the spectrum of a sinewave of the frequency of 200 Hz can be seen, the sampling of which is scheduled by a software timer component that is shared between more modules. The signal source was the external loudspeaker connected to the signal generator (see Fig. 1), and the sampling was performed by a mote that sent the samples over the base station to the PC, where data are processed. It can be seen that the spectrum is distorted; instead of a definite line (that is expected as the spectrum of a sinewave) the spectrum line is spread, since the sampling is not equidistant because of the insufficiency of the timer, and sampling is disturbed by random processes. In Fig. 3.b the timing of the sampling of the same sinewave is carried out in an interrupt routine of a hardware timer, so this timing process is disturbed only by other IT routines that are mainly short code segments, so their disturbing effect is negligible.

The sampling of the observed signal can also be disturbed by tasks that are related not only to the timing of processes. Measurements are performed in order to investigate the effects of these kind of disturbances on the observation of the signal in the following way.



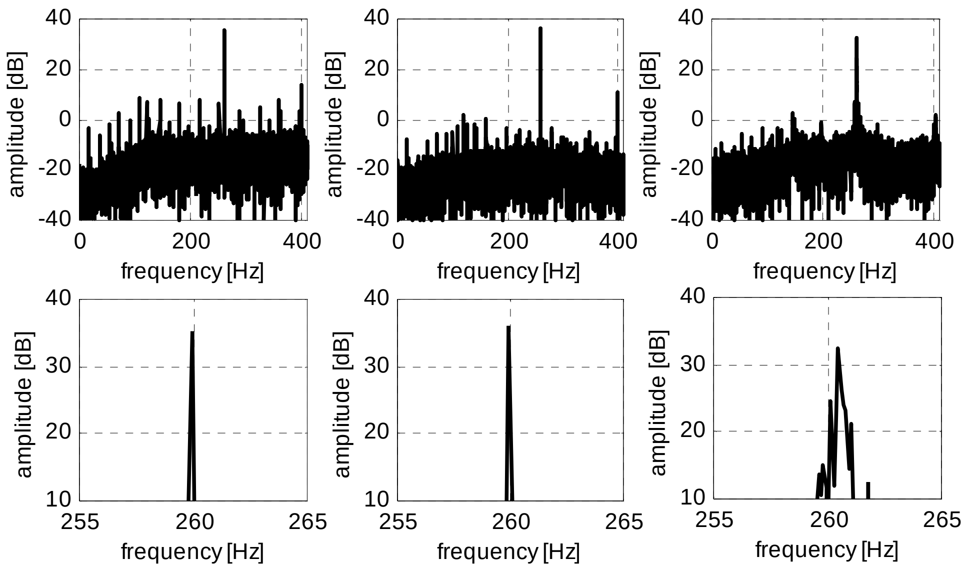
**a.**



**b.**

**Fig. 3.** Sampling with different methods. **a**: Sampling with utilization of shared timer. **b**: Sampling with utilization of single timer.

A sinewave was sampled by a mote, the sampling was controlled by a separate HW timer, and sampling was started in an event. Since the event has medium level priority it can be interrupted, so the effect of other processes can be investigated. It offers, however, satisfying quality of timing, since the processes that can interrupt events are mainly short ones. As a disturbing process the radio component was used, since it is used anyway for the transmission of the collected data. Data are sent in 25 byte size packets. For the radio component a backoff time parameter ($t_{\mathrm{backoff}}$) can be set. After the initiation of the sending of a message, the physical transmission of the message is started within a $t_{\mathrm{backoff}}$ interval with uniform distribution. With the different setting of the backoff time the spectrum of the observed sinewave was investigated. The lowest the value of the $t_{\mathrm{backoff}}$ parameter is, the more deterministic the effect of the operation of the radio component is. The spectrum of the sinewave that was sampled at different $t_{\mathrm{backoff}}$ parameter settings can be seen in Fig. 4. In the second line of the figure series the main spectrum lines can be seen. The backoff time decreases from right to left, and it is approximately 2.5 ms, 0.5 ms and 0 ms in Fig. 4.c, 4.b, 4.a respectively. According to the spectrums it can be concluded, that the effect of the disturbing processes depends on its deterministic or stochastic nature. If the behavior of the disturbance is deterministic, it influences the sampling periodically, so its effect in the spectrum appears as parasitic spectrum lines, as it can be seen in Fig. 4.a. Since the modification of the original sampling time instants is a nonlinear operation, it can be interpreted as the "mixing" of the frequency of the observed signal ($f_{\mathrm{o}}$) and the frequency of the disturbance ($f_{\mathrm{d}}$). Hence, the spurious spectrum lines appear mainly on the frequencies that are the linear combination of the $f_{\mathrm{o}}$ and $f_{\mathrm{d}}$. The random disturbances—that are produced with higher backoff interval—cause the disappearing of spurious spectrum lines, the spreading of the spectrum lines of the original sampled signal and the increasing of the noise [10], as it can be seen in Fig. 4.c. In Fig. 4.b the spurious spectrum lines has lower amplitude than in Fig. 4.a, since the disturbance of radio is no more periodic; the main spectrum line is, however, sharper than that in the Fig. 4.c.



**a.** **b.** **c.**

**Fig. 4.** Effect of the background tasks on sampling. **a**: Periodic disturbance: period=$T_{\mathrm{p}}$=28ms, $t_{\mathrm{backoff}}$ =0 ms. **b**: average period=$T_{\mathrm{p}}$, $t_{\mathrm{backoff}}$ =0.5 ms. **c**: average period=$T_{\mathrm{p}}$, $t_{\mathrm{backoff}}$ =2.5 ms.

## V. Synchronization

### *A. General Issues of Synchronization*

A crucial task in the system is the synchronization, because there are many autonomous subsystems (each mote and the DSP). Synchronization has key importance because of the strict stability condition referring to the knowledge of the feedback path.

If the sampling on motes and processing of the sampled data on the DSP occurred independently of each other, the delay between the sampling and the processing of the signal would vary at least in a one sampling period interval, as it can be seen in Fig. 5. In the figure the sampling and processing times are signed with vertical lines on the time axes, and it is assumed that the data transmission time $T_t$ is constant and the DSP processes the most currently received data. Constant $T_t$ is ensured by the deterministic network protocol.

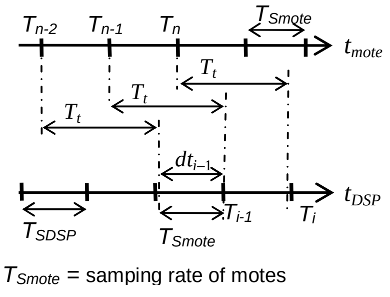


**Fig. 5.** Unsynchronized units

In $T_i$ the delay between the processing and sampling ($T_i$ and $T_n$ respectively) is about $T_t$, since the data arrives directly before $T_i$. However, in the processing time $T_{i-1}$ the delay is ($T_t + T_{Smote}$), since the data sampled in $T_{n-1}$ arrives slightly after $T_{i-1}$ so the DSP processes the previous data that has arrived approximately $T_{Smote}$ time before $T_{i-1}$. This example shows the two extreme values of the delay, but it can change anywhere within the [$T_t$…$T_t$+$T_{Smote}$] interval. Since $\mathbf{A}(z)$ includes this delay, it changes also continuously during the operation so it differs from its identified value, hence $\mathbf{W}(z)$ is no more optimal. Moreover it can occur that $\mathbf{W}(z)$ doesn't satisfy even the stability condition. This is the simplest case, when the DSP processes the most current data, but more delay can be introduced in other processing methods. In many applications the main goal is to hold the delay on constant level, since constant delay can be taken into account at the design phase, but changing delay alters the original conditions.

For the synchronization there are different basic approaches [11]. Units can be synchronized by matching the time instants of sampling and signal processing physically to each other, which requires the tuning of the clock (i.e. scheduler) of the units.

Another approach for the synchronization is the interpolation, when the effect of unaligned sampling and processing time instants are compensated by computation. By means of interpolation the exact value of the signal in the processing time instants can be estimated, so eliminating the changing delay. Interpolation can be realized on the motes or on the DSP. In the first case the DSP acquires asynchronously data, and motes calculate the signal value at the request time instants. In the second case the motes send data asynchronously and the DSP estimates the value of the signal in the processing points according to the previous samples. In the first case the computation is distributed in the network, but the transmission of request messages means extra load for the network. In the second case the interpolation demands high computation load on the DSP, since the data from every mote should be handled individually.

The interpolation can also mean the interpolation of the algorithms' parameters. Let's take the example when the data $x$ arriving from the network is multiplied by a function $f(t)$. In the case when no synchronization is applied, the input signal $x(T_n)$ (sampled in $T_n$ in Fig. 5) would be multiplied in the processing time $T_i$ by $f(T_i)$. With a synchronized algorithm, however, $x(T_n)$ is multiplied by $f(T_i–dt)$, where $dt$ is the time difference between the arrival time of $x(T_n)$ and $T_i$, so the changing delay $dt$ is compensated.

In our system the following synchronization method is applied. On the motes the sampling instants are synchronized (sampling occurs on each mote at the same time) and the DSP uses linear interpolation for fitting the data arrived from WSN to its processing time. This is a convenient solution, since the data arrived from the motes can be handled uniformly in the interpolating method—the same parameters can be used in the interpolation that can be performed for each mote at the same time—since they are sampled by motes synchronously. The physical synchronization of the whole system is not possible, since the sampling frequency of the DSP is fixed, and in some cases the sampling frequency on the motes can not reach that of the DSP. Interpolation on motes would require extra information messages that are sent from the DSP to the motes, which would cause extra network traffic.

## *B. Synchronization in Sensor Network*

Since the synchronization of motes has key importance, it is presented in detail. A PLL like method (see Fig. 6) was worked out that requires a reference mote the sampling rate of which the other motes are synchronized to. In the timers of the motes a counter runs with the clock frequency of the motes. When it reaches its programmed maximal value ($N_{\mathrm{div}}$) it will be cleared, and an interrupt is generated where the sampling of microphone's signal occurs. The value $N_{\mathrm{div}}$ determines the sampling frequency: $f_s = T_s^{-1} = f_{\mathrm{quartz}} / N_{\mathrm{div}}$ (where $T_s$ is the sampling interval). The time function of the value of the counter is a sawtooth signal, at the falling edge of which the sampling is carried out. The synchronization of the sampling frequency on motes is reached by holding the phase difference between the sawtooth signals constant with the structure that can be seen in Fig. 6. The reference mote sends synchronization messages at the sampling time instants. At the reception time of these messages the motes read the value of the counter of their timer that is $N_l$. Since the value of the sawtooth signal is proportional with its phase, this sampling and hold operation is analogous with the phase detector function, so with the tuning of the sampling frequency (changing $N_{\mathrm{div}}$) the phase difference between the sawtooth signal on the reference and on the other motes ($N_l$) can be held constant as shown in Fig. 7 (solid line).

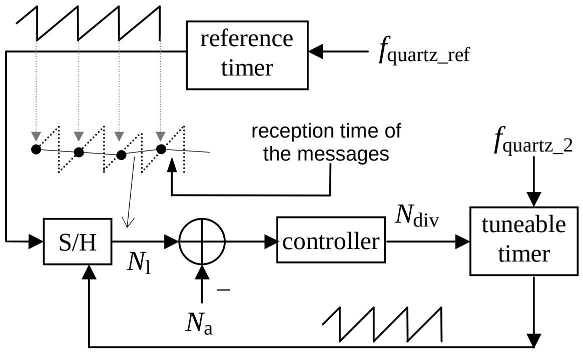


**Fig. 6.** Block diagram of synchronization algorithm

If the desired value of $N_l$ is $N_a$, then the following algorithm has to be followed: if $N_l < N_a$, $f_s$ should be increased ($N_{\mathrm{div}}$ should be decreased), otherwise $f_s$ should be decreased. If no interaction is taken place this phase changes continuously—as shown in Fig. 7 (dashed line)—because of the frequency error of the clock generators of the motes. Since the maximal value of the counter of the timer is $N_{\mathrm{div}}$, the maximal output value of the phase detector is also $N_{\mathrm{div}}$ that is in our case 4096. Since the synchronization messages carry information only on their time of arrival, any data messages can be used for this purpose, the only constrain is, that it should be sent by the reference mote at predefined points. It means, that synchronization of the sampling in this system means no extra network traffic, since it is realized with data messages that should be sent anyway. The synchronization needn't to be performed in every sampling period, because the clock of motes are accurate enough to stay near (some microsecond) to the synchronized state for longer time (cca. 1 sec).

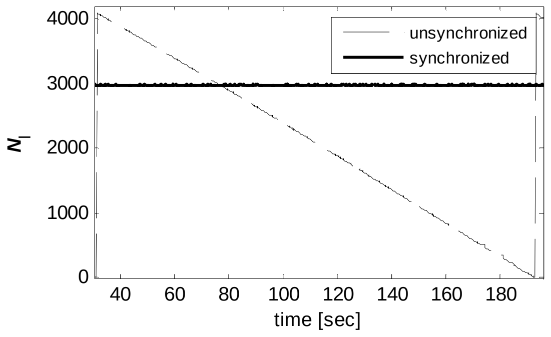


**Fig. 7.** Synchronization measurement results

*C. Synchronization on the DSP*

The DSP uses linear interpolation. It means, that the DSP measures the time $dt_k$ and in the processing time $T_k$ (see Fig. 5) it uses in the signal processing algorithm the interpolated value of the input signal that is $\hat{f}(dt_k)$. The interpolation is carried out according to Fig. 8. $d_2$ is the last and $d_1$ is the next value of the input signal. Since $d_1$ is not known in $T_k$, one sample delay has to be introduced. A possible form of the interpolation is (4):

$$\hat{f}(dt) = d_1 \frac{dt}{T_{Smote}} + d_2\left(1 - \frac{dt}{T_{Smote}}\right) \quad (4)$$

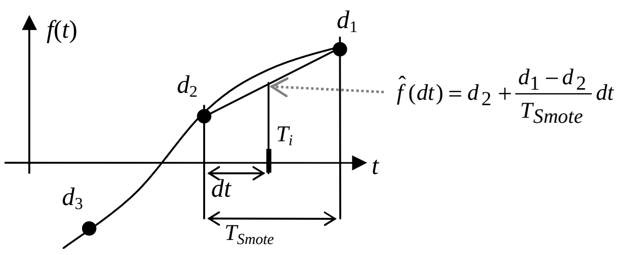


**Fig. 8.** Linear interpolation

It can be seen that the interpolation can be interpreted as a FIR (finite impulse response) filtering, where the filter coefficients are calculated by using *dt*. The interpolation can be carried out by using the Lagrange-polynomials: in an *n* order interpolation a polynomial of order *n* is fitted on the (*n*+1) samples, and according to this polynomial the value of the fitted function can be estimated also between the sampling points, where the exact value is known. In the case of linear interpolation first order Lagrange-polynomials are used.

The order of interpolation is chosen according to trade-offs between computational demands, delay and accuracy. Higher order interpolation requires higher computational capacity since the coefficients of the FIR filter has to be calculated real-time according to *dt*. Higher order interpolation uses more samples from the past, so increases the delay, which suggests the utilization of lower order interpolation, since in a control loop delay reduces the dynamics of the system. In Fig. 9 and 10 the properties of first and second order interpolation is compared since these two methods has the lowest delay. The calculation of second order interpolation is carried out as follows:

$$\hat{f}(dt) = 0.5(1+a)ad_1 + (1-a^2)\, d_1 + 0.5(1-a)ad_3 \quad (5)$$

where $a=dt/T_{Smote}$. The transfer functions are calculated using different *dt* values. In the figures *dt* is interpreted relatively to the unit sampling interval $T_s$. According to the magnitude response, the second order interpolation has better properties since it has nearly unit gain also on higher frequencies. In Fig. 9 the curves that belong to $dt_j=1-dt_i$ parameter pairs are overlapped, since the absolute value of (4) is symmetrical for these pairs.

The delay is calculated by equation $(-\varphi(\vartheta)/\vartheta)$, where $\varphi(\vartheta)$ is the phase characteristics of the interpolation and $\vartheta$ is the frequency relative to the sampling frequency. The delay vs. frequency plot shows that on low frequency each interpolation provides the appropriate –*dt* delay, so they compensates the *dt* delay in Fig. 8 and Fig. 5. In the case of second order interpolation the delay characteristics is not symmetric, since it uses two samples from past ($d_2$ and $d_3$) and one ($d_1$) from the future (naturally introducing the one sample extra delay for ensuring the causality), thus in some cases it provides worst delay compensation (*dt*=0.6). Due to this asymmetry it is recommended to use odd order interpolation. Because of these facts first order interpolation was chosen in our system, since according to practical tests it provides enough accuracy in estimation, but requires minimal computational capacity. The inaccuracy of the interpolation in terms both the delay compensation and amplitude estimation decreases on higher frequencies, so it is recommended not to use the full bandwidth that would be allowed by the sampling frequency.

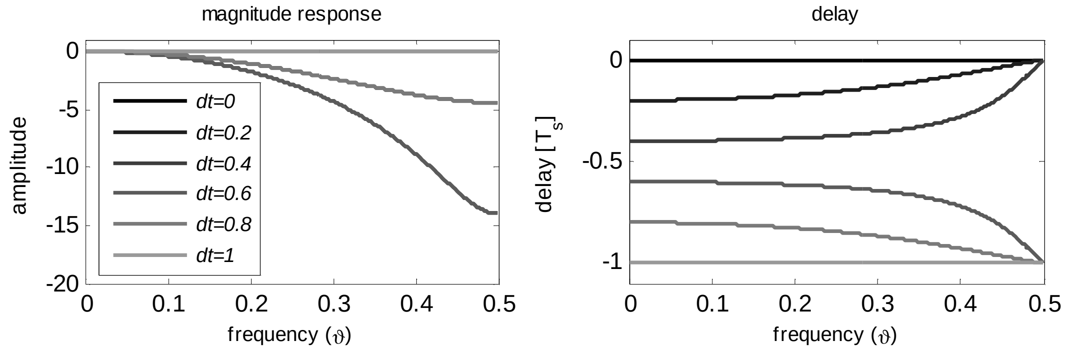


**Fig. 9.** Transfer function and delay of linear interpolation

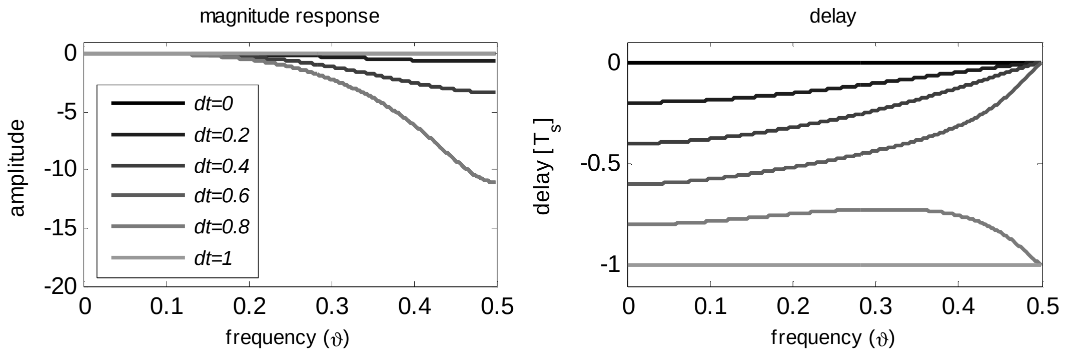


**Fig. 10.** Transfer function and delay of linear second order interpolation

## VI. Data Transmission And Signal Processing Methods

### *A. Simple Signal Collecting Network*

In the basic configuration the wireless sensor network performs the sampling of the noise to be suppressed, and the sensors send the sampled value to the DSP over a gateway mote. Data are transmitted in 25 sample size radio packets, since beyond the overheads this is the maximal size of a data block, so the ratio of the overhead that belong to each packet can be minimized. This makes the utilization of the radio channel more efficient.

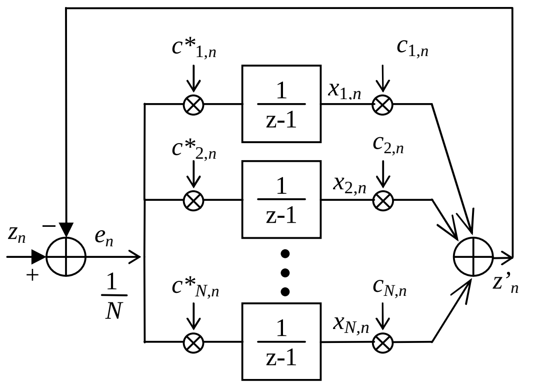


**Fig. 11.** Block diagram of recursive Fourier-analyzer

The noise control algorithm is a resonator based one [12]. This algorithm is optimized for the suppression of periodic acoustic disturbances. The basic building block of the system is a resonator based recursive Fourier-analyzer (FA) the block diagram of which can be seen in Fig. 11. The system is described by the state equations (6) and (7)

$$x_{i,n+1} = x_{i,n} + {}^{1}/_{N} \cdot c^*_{i,n} \cdot e_n \tag{6}$$

$$z'_n = \sum_{i=1}^{N} x_{i,n} c_{i,n}; \quad c_{i,n} = e^{j \cdot 2 \cdot \pi \cdot i \cdot f_1 \cdot n} \tag{7}$$

where $n$ is the time index, $f_1$ is the fundamental harmonic frequency and $N$ denotes the number of the Fourier-coefficients. The complex basis functions that belong to the $i$-th harmonic frequency are denoted by $c_{i,n}$. The structure is appropriate for recursive calculation of Fourier-coefficients of periodic signals. The state variables $x_i$ are the Fourier-coefficients: (7). The operation can be explained as follows. The $i$-th resonator channel has infinite gain on the $i$-th harmonic frequency $f_i$, since the input signal of the frequency of $f_i$ is mixed down to DC by multiplying by $c^*_{i,n}$ where the integrator has infinite gain. Since this signal is mixed up again by $c_{i,n}$ this means that infinite gain is pushed up to $f_i$. Since the resonator is in a negative feedback loop, the infinite gain means that the error signal on the frequency $f_i$ is zero.

In the noise control system the signal of each sensor is fed to a FA block (these data are denoted by $z_n$ in Fig. 11), so the noise signals are decomposed into Fourier-coefficients. Let's denote by $x_i^j$ the $i$-th harmonic Fourier-coefficient that belongs to the $j$-th input (sensor). The output signal is calculated according to (8) and (9):

$$q_{i,n+1}^k = q_{i,n+1}^k + \mu \sum_{l=1}^{L} w_{k,l}(z_i) x_{i,n+1}^l; \quad k=1 \ldots K \tag{8}$$

$$y_n^k = \sum_{i=1}^{N} q_{i,n}^k c_{i,n}; \quad k=1 \ldots K \tag{9}$$

where $w_{k,l}(z_i)$ denotes the $k$-th row and $l$-th column of $\mathbf{W}(z_i)$, $L$ and $K$ denote the number of the inputs and outputs of the system respectively. $q_i^k$ is the $i$-th harmonic Fourier-coefficient that belongs to the $k$-th output. $y_n^k$ is the signal on the $k$-th output (the input signal of the $k$-th loudspeaker in Fig. 1). This algorithm can be interpreted as an integrator type controller built on the input Fourier-coefficients with zero reference: the input (i.e. remaining) noise has to be zero. The parameter $\mu$ is the time constant of the integration, and determines the dynamic behaviour of the system. Since the control signals $y_n^k$ got to the sensors through a transfer function matrix $\mathbf{A}(z_i)$, the input signals $x_i^j$ are coupled through the matrix $\mathbf{W}(z_i)$ to the output. $\mathbf{W}(z_i)$ ensures the negative feedback that is required for the stability.

The synchronization of the DSP to the motes is basically performed by linear interpolation, but the interpolation of the whole FA structure to the motes was also tested. In this case the value of $c_{i,n}$ in (6) and (7) is calculated for the time when $z_n$ signals arrive from the motes.

The basic problem of this system emerges, when the utilization of lots of sensors is required, since the bandwidth of the radio network allows the transmission of the data of about 3-4 sensors at the sampling frequency of cca. 2 kHz. The number of the microphones in some systems might be about of the order of ten, so it is advantageous to work out methods for the increasing of the number of the sensors. The large number of sensors is required because the noise suppression is restricted to a limited surrounding of the microphones, so for appropriate noise cancellation in large space more sensors are required.

### *B. Data Reduction with Distributed Signal Processing*

One possible way for the decreasing of the amount of data to be sent over the network is the pre-processing of the signals and the transmission of only the signal parameters required for the control. In our system it is solved by the Fourier-decomposition of the signals right on the motes. The structure of this kind of ANC system can be seen in Fig. 12.

Since in the control algorithm (8) only the Fourier-coefficients are required, this solution causes no change in the functionality. Since these coefficients change slower than the signal itself, lower transmission rate is allowed, so the limitation of bandwidth is less relevant in terms of number of noise sensing motes. This makes possible the expansion the number of motes without decreasing the sampling frequency. A trade-off is, however, necessary in the number of sensors, since the higher the number of the sensors is the longer the time is that is required for the transmission of their data. This reduces the dynamic of the system, so makes the control slower.

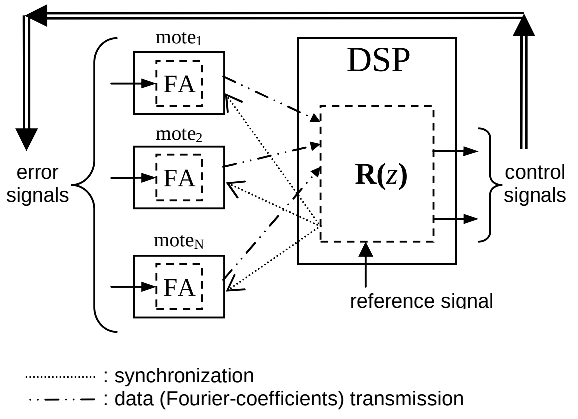


**Fig. 12.** ANC system with distributed Fourier-analyzer

The bottleneck of the system in this case is the computational capacity of the motes, since the calculation of the FA structure poses high computational demand on the motes.

In this case beyond the synchronous sampling in WSN also another synchronization problem emerges: the consistency of the base functions has to be ensured in the whole system (on each mote and on the DSP), since the phase value of the Fourier-coefficients can be interpreted only by maintaining a global reference (the base functions) in the system. It is solved with the continuous transmission of the phase and frequency of the base functions (complex exponential functions), but synchronization of sampling on motes is a necessary condition.

## VII. Conlusions

In this paper the introduction of some aspects of the design of a wireless active noise control system was addressed. The two most important issues in connection with the signal observation required for the control algorithm are the sampling of the controlled signal and the synchronization of the sampling. In the case of the sampling of signals the effects of different kind of disturbances were investigated. The constrained resource of sensor network in terms of the bandwidth of the radio channel requires special solutions in the data transmission method in order to make the system suitable for the integration of more sensors. These methods reduce the amount of data to be transmitted by the sensors, so make the bandwidth limit less relevant.

The main goal in the future is the research of the possibilities of the extension of the number of sensors without the degradation of the performance.

## References

[1.] Mathiesen, M., G. Thonet, N. Aakwaag, „Wireless ad-hoc networks for industrial automation: current trends and future prospects," *Proceedings of the IFAC World Congress*, Prague, Czech Republic, July 4-8, 2005

[2.] Akyildiz, I. F., W. Su, Y. Sankarasubramaniam, and E. Cayirci, „Wireless sensor networks: A survey," *Comput. Netw.*, vol. 38, no. 4, pp. 393-422, 2002

[3.] Kuo, S. M., D. R. Morgan, „Active Noise Control: A Tutorial Review," *Proceedings of the IEEE*, vol. 87. No. 6., pp. 943-973, June. 1999.

[4.] Orosz, Gy., L. Sujbert, G. Péceli, „Testbed for Wireless Adaptive Signal Processing Systems," *Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, Warsaw, Poland, May 1-3, 2007

[5.] URL: www.analog.com/UploadedFiles/Associated_Docs/ 50400613ADSP_21364_EZ_KIT_Lite_Manual_Rev_2.0.pdf

[6.] URL: www.xbow.com/Support/Support_pdf_files/ MPR-MIB_Series_Users_Manual.pdf

[7.] Gay, D., P. Levis, D. Culler, E. Brewer, *nesC 1.1 Language Reference Manual*, May 2003

[8.] L. Sujbert, „A new filtered LMS algorithm for active noise control," *Proceedings of the Active '99 - The International EAA Symposium on Active Control of Sound and Vibration*, Dec. 1999, Fort Lauderdale, Florida, USA, pp. 1101-1110.

[9.] Elliott., S. J., P. A. Nelson, „The Behavior of a multiple Channel Active Control System", *IEEE Trans. on Signal Processing*, Vol. 40, No. 5, May 1992.

[10.] Brannon, B., „Aperture Uncertainty and ADC System Performance," *Application Note AN-501, Analog Devices*

[11.] Molnár, K., L. Sujbert, G. Péceli: „Synchronisation of sampling in distributed signal processing systems," *Int Symp. On Intelligent Signal Processing, WISP 2003.*, Budapest, Hungary, sept. 2003.

[12.] Sujbert, L., G. Péceli, „Periodic noise cancellation using resonator based controller," *1997 Int. Symp. on Active Control of Sound and Vibration, ACTIVE '97*, pp. 905-916, Budapest, Hungary, Aug. 1997.