Energy-Saving Forecasting Techniques for Measurement Data Transmitting WSNs

Florian Strakosch and Faouzi Derbel
Leipzig University of Applied Sciences, Leipzig, Germany
Email: Florian.Strakosch@htwk-leipzig.de

***Abstract*—One of the most important reasons for the usage of relatively inexpensive energy-self-sufficient wireless sensor networks is their high reliability. Research in this particular field has been intensified recently. However, most of the proposed approaches have one problem in common: lifetime reduction due to an increased demand on data transmission or packet length. In this paper we propose a new technique for almost reversing the effects of this conflict by system identification and forecasting. Our work focuses on designing a close-to-perfect model for each sensor type and to use this for measurement data prediction. Then a data transmission is required only when the difference between the measured and predicted value exceeds a certain threshold. If, for example, every second prediction is true in this context, the lifetime of a wireless sensor node can be extended by more than 45% while still constantly presenting accurate values to the user.**

***Index Terms*—wireless sensor networks, system identification, forecasting, prediction, ARX, Kalman, energy-saving, lifetime extension**

I. Introduction

Nowadays manufacturers in general are faced with the problem of constantly changing market needs. New designs emerge, costumer interests shift and regulations are changed ever quicker. This in return is forcing engineers worldwide to adjust the production lines more often.

The factory automation itself was invented for such conditions and can be re-engineered within a couple of hours at best. In terms of construction, wiring in particular, the situation is significantly more complicated as well as costly.

Energy-self-sufficient wireless sensor networks are meant to fill this gap. The advantages of using the new sensing elements are obvious: they are small, light, inexpensive and do not need any wired connection. Hence, they can be mounted almost everywhere and relocated (or even replaced) easily with a minimum of necessary engineering. Nevertheless, there are also some new challenges in comparison to conventional sensors such as the need for self-organization and routing in ever changing environments as well as issues of the questions about accuracy and availability. In other words: wireless sensor nodes should at least fulfill the same requirements wired ones do.

Researchers all over the world are working on this topics and similar problems, making wireless sensor networks even more attractive to customers. Yet, one major issue still remains: in order to really be autarkic, the limited available energy and therefore the limited lifetime of the nodes has to be considered. The only way for treating this problem is by reducing the time the node is awake or transmitting data or both.

In this paper we will consider a decreased amount of sending intervals by forecasting and comparing predicted and measured values to decide whether a transmission is necessary or not.

II. State of the Art and Related Works

Based on new trends in smart diagnostic especially for dwelling buildings, the idea was to rather read out meters for electricity, heating, water and so on by radio transmission then by sending a workman with pencil and paper to every single apartment. To make it even more interesting and "designing" an advantage to standard components for potential customers the option of presenting "live" data in addition to the annual reading was implemented. Further, in terms of building automation and climate control, temperature as well as humidity sensors were meant to be added for comfort (and data forwarding) reasons. By doing so the whole idea became interesting for the industry, too.

Now the dimensions of the sensors do not allow for large batteries. Hence, energy efficiency was the main goal. Some of the solutions were larger measurement intervals, sampling data and sending them in one big packet instead of several small ones or just transmitting changes. As any compromise they had disadvantages like loss of information or not presenting "live" data at the exact time. Again, new implementations were needed.

One approach [1] was to use system identification and forecasting for saving energy, for example. The idea behind that work was to design a system model from given physical coherences and using this model for predicting r upcoming values. Afterwards, the results were send in one big packet to the gateway. The process repeated after r time steps or in case of the difference between a measured and predicted value exceeded a given limit. Hence, "live" data were displayed at the correct time but the filter chosen was very complex and the issue with the packet size still remained.

III. Forecasting Technique

The approach presented in this paper differs from the one given above in three ways:

- the physical model is considered unknown (black box)

- only one value is to be transmitted every r time steps
- prediction runs parallel on sensor nodes and gateway

The first point implies that the only information given are the output value and the sampling rate, which requires a system identification process out of sample data. This needs to be done offline in advance to the regular forecasting and with the help of the methods described in the following chapter.

Declaration two and three are more or less leading to each other. As shown in figure 1 the idea of our approach is based on the sensor (almost) simultaneously measuring and predicting the next value. Afterwards, both are compared against each other concerning a tight threshold. Only when the limit is exceeded or after r time steps a new transmission is initiated, which also is to be interpreted as a sign of life.

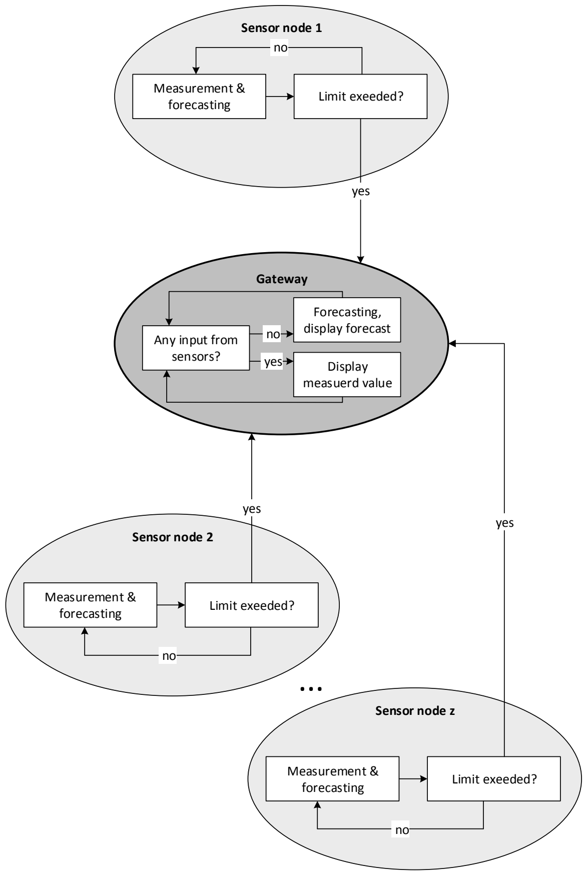

Fig. 1. Forecasting algorithm on gateway and sensor nodes.

Meanwhile, the gateway is doing nothing but predicting values using the very same algorithm and presenting the results to a human machine interface (HMI). As long as there is no input coming from a sensor and the r time steps have not passed, the gateway will assume that the forecasting is good and keeps displaying the results. In case there is no input for more then r time steps, the gateway is still able to continuously present predicted values to the HMI but will inform the system and / or the user about a possibly broken connection.

On the other side the gateway needs to know the correct parameters of each sensor to perform an accurate forecasting. This will be assured by the sensor nodes sending it's parameters as well as the first r measured values during initialization phase.

IV. System identification - Basics

As mentioned above, increasing measurement intervals or sending sets of sampled values are not appropriate methods. Usually, the user expects some data constantly displayed on his screen. To address this demand, using forecasting parallel on sensor and gateway, as presented in figure 1, is a promising idea. Therefore, an accurate system identification for parameter estimation implying a priori knowledge is needed. Hence, the first step is collecting reasonable amounts of sample data, ideally representing the whole process including all maxima and minima. These data should preferably be input, output and white noise.

Due to having access to the output values and the sampling rate only, some forecasting algorithms will not work at all and others are too complex for being handled by relatively inexpensive wireless sensor nodes. This immediately suggests the usage of simple time domain models. The most promising approaches are presented in detail now:

A. ARX-Model

The AutoRegressive model with eXogenous input (ARX) is based on the linear difference equation describing the relationship between input $u(t)$ and output $y(t)$ at time t:

$$\begin{aligned} y(t) + a_1 \cdot y(t-1) + \cdots + a_n \cdot y(t-n) \\ = b_1 \cdot u(t-1) + \cdots + b_m \cdot u(t-m) \end{aligned} \tag{1}$$

Thereby discrete time is assumed because of most data being collected by sampling. Also the influence of (white) noise is neglected for complexity reasons.

Since the major goal of this approach is forecasting, which means determining the next output value from previous observations, a more appropriated way of looking at (1) is:

$$\begin{aligned} y(t) = -a_1 \cdot y(t-1) - \cdots - a_n \cdot y(t-n) \\ +b_1 \cdot u(t-1) + \cdots + b_m \cdot u(t-m) \end{aligned} \tag{2}$$

As mentioned before, the only accessible values in case of wireless sensor nodes are output and sampling rate. Hence, all $u(t)$ terms are unknown and therefore could be assumed to equal $y(t)$. Following this idea (2) would become:

$$y(t) = -a_1 \cdot b_1 \cdot y(t-1) - \cdots - a_n \cdot b_m \cdot y(t-n) \tag{3}$$

Since there is no need for two different parameters anymore, $a_n \cdot b_m$ can be substituted by a_n, which leads to the following equation:

$$y(t) = -a_1 \cdot y(t-1) - \cdots - a_n \cdot y(t-n) \tag{4}$$

It can be noted in a more compact way by introducing the vectors

$$\begin{aligned} \boldsymbol{\theta} &= \left[a_1 \ldots a_n\right]^T && (5) \\ \boldsymbol{\phi}(t) &= \left[-y(t-1) \cdots - y(t-n)\right]^T && (6) \end{aligned}$$

as

$$y(t) = \boldsymbol{\phi}^T(t) \cdot \boldsymbol{\theta} \tag{7}$$

where $\boldsymbol{\theta}$ contains the unknown weights (parameters) of the linear difference equation. That is when the previously collected sampling data over a time interval $1 \leq t \leq N$ are called into action. Using them according to [1] and [2], $\boldsymbol{\theta}$ can be determined so as to fit the $y(t)$ values as well as possible to the sample data. Therefor, the least squares method is used such as:

$$\boldsymbol{\theta}_N = \{\sum_{t=1}^{N}(\boldsymbol{\phi}(t) \cdot \boldsymbol{\phi}^T(t))\}^{-1} \cdot \sum_{t=1}^{N}(\boldsymbol{\phi}(t) \cdot y(t)) \tag{8}$$

This linear problem can be solved easily with the help of modern numerical software.

B. ARMA-Model

The ARMA-Model consists of two parts, an autoregressive component (AR) weighting the previous output values added by an error correction term

$$y_{\mathrm{AR}}(t) = a_1 \cdot y(t-1) + \cdots + a_n \cdot y(t-n) + e(t) \tag{9}$$

and a moving average (MA) of the measured variable over time including another error correction

$$y_{\mathrm{MA}}(t) = b_1 \cdot e(t-1) + \cdots + b_m \cdot e(t-m) \tag{10}$$

The combination can be noted as follows:

$$\begin{aligned} y(t) = a_1 \cdot y(t-1) + \cdots + a_n \cdot y(t-n) + e(t) \\ +b_1 \cdot e(t-1) + \cdots + b_m \cdot e(t-m) \end{aligned} \tag{11}$$

Taking into account the condition of looking at a black box, there is no way of getting any accurate information about error terms or input values out of the system. This in return dramatically reduces (11) to

$$y(t) = a_1 \cdot y(t-1) + \cdots + a_n \cdot y(t-n) \tag{12}$$

which is identical to (4) but the algebraic sign. This means that in this special case the ARMA-Model will deliver exactly the same results as the ARX-Model. Hence, it is sufficient to use the previous discussed model for further observations.

C. Kalman-Filter

The Kalman-Filter is used to identify a discrete-time linear state-space model for observation and forecasting purposes. As shown in [3] and [4], it can be noted as

$$\boldsymbol{x}(t) = \boldsymbol{A} \cdot \boldsymbol{x}(t-1) + \boldsymbol{B} \cdot \boldsymbol{u}(t-1) + e_x \tag{13}$$

$$y(t) = \boldsymbol{C} \cdot \boldsymbol{x}(t) + e_y \tag{14}$$

where $\boldsymbol{x}(t)$ is the state prediction n by 1 vector, $\boldsymbol{x}(t-1)$ is the prior state n by 1 vector, $\boldsymbol{u}(t-1)$ is the systems' input n by 1 vector, $y(t)$ is the (predicted) systems' output, $\boldsymbol{A}$ and $\boldsymbol{B}$ are system describing coefficient n by n matrices, $\boldsymbol{C}$ is the 1 by n correlation vector between the state of the system and the actual measurement and finally e_x and e_y are gaussian error terms. In some cases this description of an observed system can be used directly by easily filling $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ with mathematical representations of the physical system like for speed or acceleration.

Again, the systems' input $\boldsymbol{u}(t-1)$ as well as the physics and especially e_x and e_y are unknown. Hence, the error terms are set to zero and an observer introducing the Kalman Gain $\boldsymbol{K}$ as a 1 by n vector is implemented in (13) replacing $\boldsymbol{B} \cdot \boldsymbol{u}(t-1)$:

$$\boldsymbol{x}(t) = \boldsymbol{A} \cdot \boldsymbol{x}(t-1) + \boldsymbol{A} \cdot \boldsymbol{K} \cdot (y(t-1) - \boldsymbol{C} \cdot \boldsymbol{x}(t-1)) \tag{15}$$

$$y(t) = \boldsymbol{C} \cdot \boldsymbol{x}(t) \tag{16}$$

In this representation $y(t-1) - \boldsymbol{C} \cdot \boldsymbol{x}(t-1)$ is the difference between the real and the predicted measurement and $\boldsymbol{K}$ is a weighting factor. Substituting (15) into (16) one will get:

$$\begin{aligned} y(t) = \boldsymbol{C} \cdot \{\boldsymbol{A} \cdot \boldsymbol{x}(t-1) + \boldsymbol{A} \cdot \boldsymbol{K} \cdot (y(t-1) \\ -\boldsymbol{C} \cdot \boldsymbol{x}(t-1))\} \end{aligned} \tag{17}$$

or (by substituting repeatedly according to [4] and [5])

$$y(t) = \sum_{n=1}^{N}[\boldsymbol{C} \cdot \{\boldsymbol{A} \cdot (\boldsymbol{I} - \boldsymbol{K} \cdot \boldsymbol{C})\}^{n-1} \cdot \boldsymbol{A} \cdot \boldsymbol{K} \cdot y(t-n)] \tag{18}$$

When comparing (18) to (4) it is obvious that $-a_n$ equals $\boldsymbol{C} \cdot \{\boldsymbol{A} \cdot (\boldsymbol{I} - \boldsymbol{K} \cdot \boldsymbol{C})\}^{n-1} \cdot \boldsymbol{A} \cdot \boldsymbol{K}$. Hence, the parameters estimated with the help of (8) can be used for determining $\boldsymbol{A}$, $\boldsymbol{C}$ and $\boldsymbol{K}$ again.

Another approach would be using Markov-type parameters in form of $[\boldsymbol{C} \cdot \boldsymbol{A} \cdot \boldsymbol{K}, \boldsymbol{C} \cdot \{\boldsymbol{A} \cdot (\boldsymbol{I} - \boldsymbol{K} \cdot \boldsymbol{C})\} \cdot \boldsymbol{A} \cdot \boldsymbol{K}, \boldsymbol{C} \cdot \{\boldsymbol{A} \cdot (\boldsymbol{I} - \boldsymbol{K} \cdot \boldsymbol{C})\}^2 \cdot \boldsymbol{A} \cdot \boldsymbol{K}, \boldsymbol{C} \cdot \{\boldsymbol{A} \cdot (\boldsymbol{I} - \boldsymbol{K} \cdot \boldsymbol{C})\}^3 \cdot \boldsymbol{A} \cdot \boldsymbol{K}, \ldots]$.

Still, modern numerical software is needed for getting appropriate values.

V. System identification - Examples

Knowing several system identification processes for the given black box, the next step is to compare them against each other concerning different scenarios. Therefore, data sets of temperature, humidity and dew point with up to 7000 samples are used. The data has been collected from a real factory environment representing one workday and the following weekend. The constant sampling rate is 15 seconds.

To determine the necessary amount of sampling and validation data, sets of 100/100, 1000/100, 3000/1000 and 6000/1000 have been chosen and tested.

In addition, the three system models explained above have been altered concerning the number of parameters used within from $n = 1$ to 10. In other words, they ranged from 1st to 10th model order each.

With the help of the Matlab System Identification Toolbox the best fitting parameters out of the given combinations were identified for every model. Afterwards, the most promising candidates are compared using their respective forecasting algorithm and a separate set of validation data. A predicted value was counted as "true" when it did not differ by more than 1% from the measure one.

A. ARX-Model

For our examples the best compromise between complexity of the calculation and forecasting accuracy appeared to be at a set of 3000 sampling data, 1000 validation data and $n = 5$ parameters. Using temperature values together with the Matlab System Identification Toolbox (4) would become

$$\begin{aligned} y(t) = {} & 1.2669 \cdot y(t-1) - 0.4078 \cdot y(t-2) \\ & +0.1899 \cdot y(t-3) - 0.0310 \cdot y(t-4) \\ & -0.0181 \cdot y(t-5) \end{aligned} \tag{19}$$

delivering a forecasting probability of better than 80%. This means that in theory 4 out of 5 measurements do not need to be transmitted but can be reliable predicted.
Figure 2 is representing how close the prediction comes to the measurement:

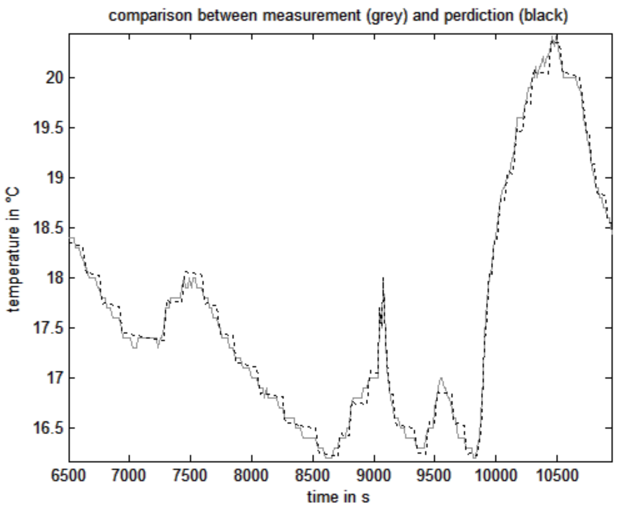

Fig. 2. Comparison between measurement (grey) and a 5th order ARX-Model based prediction (black).

B. ARMA-Model

Since the ARMA-Model can be reduced dramatically due to missing process information, (12) would look exactly like (19) and therefore giving the same results as shown in section V-A.

C. Kalman-Filter

Using the same sets of data as in V-A and the 5th model order for a better comparison, the Kalman-Filter gives slightly inferior results of about 79% forecasting probability. The differences can be seen in particular by comparing figure 2 to figure 3.
Unfortunately, the parameter handling for the Kalman-Filter is much more complex than for the ARX-Model. The system describing coefficient matrices determined by the Matlab System Identification Toolbox using (17) are:

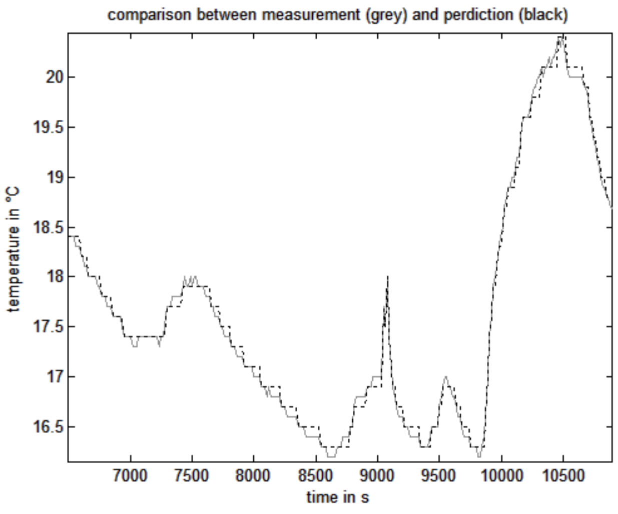

Fig. 3. Comparison between measurement (grey) and a 5th order Kalman-Filter based prediction (black).

$$\boldsymbol{A} = \begin{bmatrix} 1.0008 & -0.0171 & -0.0136 & 0.0064 & -0.0064 \\ 0.0118 & -0.1984 & 0.1143 & -0.1731 & 0.6416 \\ 0.0040 & 0.7383 & -0.4760 & 0.2917 & 0.3119 \\ -0.0107 & -0.3727 & -0.6338 & -0.0966 & 0.1101 \\ 0.0015 & 0.2691 & -0.0476 & -0.6385 & 0.0381 \end{bmatrix}$$

and

$$\boldsymbol{C} = \begin{bmatrix} 1031.4 & 3.5159 & 1.8640 & -1.0012 & 1.7989 \end{bmatrix}$$

In addition, the estimation error covariance n by n matrix $\boldsymbol{P}$ needs to be introduced to run the algorithm. Its initial value is assumed to be the system noise also determined by the Matlab System Identification Toolbox. Therefore, $\boldsymbol{P}(t-1) = 0.0601$ in our example.
According to [4] and [6], there are several ways for representing the Kalman-Filter algorithm. With the focus on a preferably simple notation as well as implementation we decided for the following one:

$$\begin{aligned} \boldsymbol{x}(t) &= \boldsymbol{A} \cdot \boldsymbol{x}(t-1) \\ \boldsymbol{P}(t) &= \boldsymbol{A} \cdot \boldsymbol{P}(t-1) \cdot \boldsymbol{A}^T + cov(\boldsymbol{x}(t)) \\ \boldsymbol{K}(t) &= \boldsymbol{P}(t) \cdot \boldsymbol{C}^T \cdot (\boldsymbol{C} \cdot \boldsymbol{P}(t) \cdot \boldsymbol{C}^T)^{-1} \\ \boldsymbol{x}(t+1) &= \boldsymbol{x}(t) + \boldsymbol{K}(t) \cdot (y(t-1) - \boldsymbol{C} \cdot \boldsymbol{x}(t)) \\ \boldsymbol{P}(t+1) &= (\boldsymbol{I} - \boldsymbol{K}(t) \cdot \boldsymbol{C}) \cdot \boldsymbol{P}(t) \\ y(t) &= \boldsymbol{C} \cdot \boldsymbol{x}(t+1) \end{aligned}$$

Hence, there is a need not only for more calculation steps and therefore more CPU time but also for a six times larger memory space compared to section V-A.
Still, due to its great forecasting probability the Kalman-Filter is worth to be considered for sensor types other then temperature, humidity or dew point.

VI. Energy Savings

Based on the theoretical capabilities of system identification and forecasting techniques in terms of reducing transmission, we shall consider the effects on the lifetime of sensor nodes. There are several ways of describing energy consumption and the corresponding saving potentials. To keep it simple, we assume that the receiving node is always on and ready for reception, which allows for looking at the sender only. Following this approach, according to [7], the energy consumption of a wireless sensor node can be noted as

$$E = U \cdot I_0 \cdot t_0 + U \cdot I_{\text{tx}} \cdot (t_{\text{oh}} + p_{\text{data}} \cdot t_{\text{byte}}) + (r-1) \cdot U \cdot I_{\text{CPU}} \cdot t_{\text{pred}} \quad (20)$$

Here U is the supply voltage, I_0 and t_0 are the average current and time needed for waking up, standby and going back to sleep, I_{tx} is the current drawn during transmission, t_{oh} and t_{byte} represents how long the transmission of the overhead and each byte will take while p_{data} defines the amount of data bytes that needs to be transmitted and finally I_{CPU} and t_{pred} are the current and time used for r forecasting steps by the microcontroller.

If the prediction fulfills the requirement of avoiding $r-1$ out of r transmission, the normalized energy saving can be defined as

$$\Delta E(r) = \frac{r \cdot E_{\text{standard}-\text{transmission}} - E_{\text{predicted}-\text{transmission}}}{r \cdot E_{\text{standard}-\text{transmission}}}$$

or

$$\Delta E(r) = \quad 1 - \frac{I_0 \cdot t_0 + I_{\text{tx}} \cdot (t_{\text{oh}} + p_{\text{data}} \cdot t_{\text{byte}}) + (r-1) \cdot I_{\text{CPU}} \cdot t_{\text{pred}}}{r \cdot \{I_0 \cdot t_0 + I_{\text{tx}} \cdot (t_{\text{oh}} + p_{\text{data}} \cdot t_{\text{byte}})\}} \quad (21)$$

Obviously, $\Delta E(r)$ needs to be > 0 in order to save energy, which leads to the condition $r > 1$.

To get a more comparable formula a 5th order ARX-Model has been implemented on a Crossbow's TelosB as an example. Therefore, $I_0 \approx I_{\text{tx}} \approx 20\,\text{mA}$, $t_0 \approx 0.4\,\text{ms}$, $t_{\text{byte}} = 0.032\,\text{ms}$, $I_{\text{CPU}} \approx 2\,\text{mA}$ and $t_{\text{pred}} \approx 1\,\text{ms}$.

Let us assume that the packet overhead is 25 Bytes, a sensor value is 2 Bytes and the corresponding time stamp is 4 Bytes. In addition, t_{oh} is $0.8\,\text{ms}$ and p_{data} equals 6. In this case (21), will become

$$\Delta E(r) = \frac{(r-1) \cdot 1.292}{r \cdot 1.392} \quad (22)$$

This means that for the given example the energy saving can be directly determined by the knowledge of the amount of successful forecasting steps.

Taking section V-A into account and therefore assuming that only 1 out of 5 measurements need to be transmitted, the energy saving would almost be 75%. Hence, the lifetime of a wireless sensor node could be four times higher without any loss of data and still displaying accurate values by the time of their measurement.

Yet, also the worst-case scenario with only every second prediction not exceeding the given limits would still lead to remarkable 46% energy saving or an almost doubled lifetime.

VII. Conclusion / Prospect

In this paper a new approach for reducing the energy consumption of measurement data transmitting wireless sensor networks by system identification and forecasting has been presented. It was shown that even despite the missing knowledge about the data acquisition process like physical parameters or noise performance a remarkable energy saving potential between 46% and 75% can be reached. Hence, the lifetime of a node could be extended by up to four times while still representing accurate values at the given sampling rate.

These results are based on a TelosB sensor node implementation of a 5th order ARX-Model and temperature, humidity as well as dew point sample data from a real factory environment. The next steps of our research will include determining, whether the named algorithms are suitable for other measurement value types like acceleration, vibration and force. In this context, signal processing procedures like Fast Fourier Transform (FFT) are to be examined and the limits of our approach need to be analyzed. In addition, possibilities and restrictions of the integration of this forecasting technique in a routing protocol for building a ready-to-use multi-hop system is going to be discussed.

Acknowledgment

The authors would like to thank the ESF for funding this work as a part of the junior research group "AIS - Autarkic Intelligent Sensor Networks".

References

[1] F. Derbel, "Modeling fire detector signals by means of system identification techniques," in *IEEE Transactions on Instrumentation and Measurement*, vol. 50, Munich, Germany, 2001.

[2] L. Ljung, *System Identification - Theory for the user*, 2nd ed., T. Kailath, Ed. New Jersey, New York, USA: Prentice-Hall, Inc., 1999.

[3] J. Valasek and W. Chen, "Observer/kalmen filter identification for online system identification of aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 2, 2003.

[4] K. Brammer and G. Siffling, *Kalman-Bucy-Filter: Deterministische Beobachtung und stochastische Filterung*, 3rd ed. Munich, Germany: R. Oldenbourg Verlag, 1989.

[5] C.-W. Chen, G. Lee, and J.-N. Juang, "An improved autoregressive spectral estimation method using the kalman filter identification technique," in *Procceding of the American Control Conference*, San Francisco, California, USA, 1993.

[6] D. Simon. (2001) Kalman filtering. Cleveland, Ohio, USA.

[7] F. Derbel. (2010) Smart wireless sub-metering. Munich, Germany.