Calibration of sensors in sensor networks

Leon Kleiboer, Paul Havinga
EEMCS Faculty, University of Twente
Postbus 217
7500 AE Enschede, the Netherlands
{kleiboer, p.j.m.havinga}@ewi.utwente.nl

Abstract

In this paper we present a new approach to calibrate sensors in sensor networks in an uncontrolled environment. The proposed algorithm makes a model of the distribution of the measured quantity. This model can be used to estimate and correct the bias of the sensor. The proposed, centralized calibration algorithm is a macro-calibration algorithm which tries to improve the system response as a whole, instead of optimizing the response of an individual sensor. The algorithm decides, based on measurements to apply the calculated corrections or not. Testing shows that the algorithm can be applied on e.g. temperature sensors. Systematic measurement errors can be reduced. A combination of a few accurate (expensive) sensors, and a large amount of less accurate (cheap) sensors, can be used together with the algorithm in real life applications to improve the quality of the measurements.

1 Introduction

Consider a scenario where vegetables are stored in a cold storage. In such a scenario, it is crucial to keep the temperature constant. In order to extend the shelf-life of the vegetables, it is necessary to ensure that the cooling system is able to respond accurately to external events, e.g. opening of doors, which might result in a sudden rise of temperature. The cooling system monitors the temperature using wireless sensor nodes that are equipped with temperature sensors. When a temperature sensor is biased, the output of the sensor is always ϵ degrees too high or too low (where ϵ is constant). Temperature sensors generating biased readings could result in the cooling system switching on or off prematurely, or later than it should be, which may affect the freshness of the vegetables. In order to have an accurately functioning system it is imperative to ensure that the sensors are properly calibrated. Since our temperature sensors operate within normal environmental temperatures, in the middle of the range of the sensors, we do not deal with errors that are caused due to non-linearity. When measuring at the borders of the range of the sensors, these non-linearity errors must be taken into account. Random errors, e.g. noise or transient events, are suppressed by averaging multiple samples. By applying the proposed calibration algorithm, systematic (bias) errors can be reduced, resulting in extended shelf-life of the vegetables in the example scenario. This paper will discuss related calibration algorithms with their drawbacks, section 3 and 4. We provide a new algorithm, describe its characteristics in section 5. Simulations and real life results are stated in sections 6 and 7. Section 8 lists some future work to be carried out, a conclusion is stated in section 9.

2 Calibration

In [6] a formal definition of calibration is stated. Each device has a set of parameters, $\beta \in \Re^p$. The purpose of calibration is to choose the correct parameters for each device such that they, in conjunction with a calibration function, will translate any actual device output T into the corresponding desired output $T*$. The calibration function must therefore be of the form $T* = f(\beta, T)$.For a temperature sensor these set of parameters can be deduced by creating a controlled environment. In an absolute controlled environment the desired output $T*$ is known. Directly observing each device and building a mapping from T to $T*$ to directly optimize that device response is called micro-calibration. This can be done manually. In a Wireless Sensor Network (WSN) many temperature sensors are available, calibration relationships between two or multiple sensors can be obtained. E.g. Relative calibration, calibration of individual sensors relative to each other instead of some absolute reference. More approaches exists. The opposite of micro-calibration is macro-calibration. Macro-calibration does not optimize the accuracy of one individual sensor, but optimizes the accuracy of the whole system (the WSN). Macro calibration exploits the sensor redundancy in the WSN. Macro calibration contains three steps [6]. The first step is to parameterize individual devices and model the system accuracy as a whole using these parameters. The second step is to collect data from all the devices in the system. The third step is to optimize the accuracy of the entire system, by choosing parameters for the individual devices.

3 Related work

LaMarca et al [4] suggest to use a service robot. This service robot is equipped with calibrated sensors. The robot visits every node in the network and performs a pairwise calibration between the sensors of the service robot and the sensors of the node. This service robot runs in an office environment. Disadvantage of this approach is the intrusive and disruptive behavior of the robot. Problems might arise when reaching the nodes physically becomes more complicated (e.g. rough terrain). In [1] and [2] Bychovskiy et al proposes to use relative calibration in the first phase of their algorithm. In the second phase, the relative calibration relationships are optimized. Due to errors, calibration relationships are not globally consistent. With a Localized Consistency Maximization algorithm, calibration relationships can be improved on a local scale. Calibration relationships of short paths (nodes placed closely to each other) are trusted more than relationships of longer paths. Calibration relationships can be stored in a graph. A calibration cycle invariant was introduced; a calibration graph is consistent if and only if a convolution of calibration relationships over any cycle in the graph is a null transformation. Whitehouse and Culler [6] are using among mean- and iterative-calibration, macro-calibration in order to improve the system accuracy. They are using the MICA sensor platform connected with a mica sensor board. These boards consist of a sounder and a microphone. A total of 32 nodes are placed in $30cm$ x $30cm$ grid. When measuring the Time of Flight (TOF), the distance between two nodes can be calculated. In this scenario, two devices, the sounder and the microphone, on each node, have to be calibrated. With joint-calibration, a set of equations can be made, $d* = B_T + B_R + G_T \cdot d + G_R \cdot d$, where B_T and B_R represent startup times for oscillation (constants). G_T and G_R represents transmitter volume and receiver sensitivity, respectively and are proportional to the distance. Joint-calibration performed better than mean- and iterative calibration, due to the exploitation of the redundancy of the sensors. Results can be optimized, using a priori information. The distance between two sensors must be the same, $d^*_{i,j} = d^*_{j,i}$ and the triangular inequality, $d^*_{i,j} + d^*_{j,k} - d^*_{i,k} \geq 0$.

4 Drawback of existing calibration algorithms

There are some drawbacks of the calibration algorithms mentioned in the previous section. This section explains why they are not completely suitable for calibration in a sensor network. Manual calibration of sensors in a sensor network is a time consuming task. Which makes manual calibration of sensors in a WSN simply too expensive. Relative and absolute calibration are based on the assumption that two sensors measuring the same physical process, should produce the same desired output $T*$. For nodes placed close to each other this is an acceptable assumption. When the distance between nodes increases, nodes are not measuring the same physical process. Therefore the desired output is not the same, which makes relative and absolute calibration less appropriate in larger areas. Mean-calibration will not work either. Assume that cooling-water (of e.g. a factory) has to be measured. The temperature directly at the drain will be higher than the temperature at a certain distance from the drain. The mean temperature can of course be calculated, but what is the meaning of this value? The described joint-calibration algorithm works for measuring the distance between two nodes, because a single measurement is a function of two nodes. Measurements between two nodes can be optimized due to known a priori information, like the triangular inequality. A priori information of this nature however is not available in the case of temperature measurements. A temperature measurement is a single and independent measurement.

5 Algorithm

A new calibration algorithm for temperature sensors in WSNs has to be designed, keeping in mind some properties of known calibration algorithms. As stated earlier, the main idea of the algorithm is to make a model of the distribution of the temperature. This idea is based on fact that there must be some sort of correlation between the measured temperatures by two or more nodes placed close to each other. Due to this correlation, the distribution of the temperature has a smooth shape. This shape can be flat, with a slope, with a curve or some sort of sine-shape. Sharp edges, like step-functions are unlikely. Lets define the problem of biased temperatures a little bit more. When measuring the temperature, one wants to know the exact physical temperature (at the position of sensor i), this temperature is called T^*_i. Due to an unknown bias error ϵ_i the measured temperature T_i is only known, but the physical temperature T^*_i is not known. The goal is to estimate a temperature $\hat{T}_i$ such that the difference between the physical temperature and the estimated temperature is as small as possible and in the ideal case equal to zero ($T^*_i - \hat{T}_i$=0). Estimating the temperature can be done by estimating the bias error ϵ_i of the sensor. When subtracting the estimated bias error λ_i from the measured temperature, an estimation of the temperature is obtained, $\hat{T}_i = T_i - \lambda_i$. $\hat{T}_i$ is estimated with curve fitting.

T^*_i	real physical temperature.
ϵ_i	bias error.
T_i	measured temp. $T_i = T^*_i + \epsilon_i$.
$\hat{T}_i$	estimated temperature.
λ_i	diff. measured and estimated temp.

The assumption is made that in a set of sensors the bias will be normal (Gaussian) distributed, $\mathbf{N}(\mu, \sigma^2)$ with mean μ and variance σ^2.

The goal of macro-calibration is to improve the system accuracy as a whole. Every temperature sensor, as stated earlier, i has a bias error, ϵ_i. Let's define the current system accuracy, R_{sys} as the summation of those bias errors ϵ_i, where N is the total number of nodes.

$$R_{sys} = \sum_{i=1}^{N} |\epsilon_i| \tag{1}$$

The accuracy of the system is improved if and only if the following equation holds, where λ_i is the correction for sensor i.

$$\sum_{i=1}^{N} |\epsilon_i - \lambda_i| < R_{sys} \tag{2}$$

Ideally the left side of the equation is zero, the error of each sensor is than reduced to zero and therefore also the accuracy of the system as a whole. Note that the system has a list of triples (i, x, y). Where i is the node number, x and y are the positions belonging to node i. Both notations are used in this thesis. So ϵ_i is $\epsilon_{(x_i,y_i)}$.

5.1 Phase one, parameterize individual devices

The first phase of the algorithm is to parameterize individual devices and the system as a whole. The sensor nodes are placed in a grid or placed randomly, as stated earlier. The positions of the nodes can be stored in a database. Position information is needed to calculate the corrections of the biases. This phase can also be used if a priori calibration parameters are available. If these parameters are not available it can be skipped.

5.2 Phase two, data gathering

The second phase of the macro-calibration algorithm is data gathering. Lets define an epoch. The duration of one epoch is constant (e.g. one minute) and is the same for all sensor nodes. An epoch is needed because temperature measurements must be compared. It is possible to compare temperature measurements of different sensors at the same epoch. Per epoch, several samples are taken. These samples are averaged and the averaged value of each temperature sensor is sent to a central database. Averaging is needed because two consecutive temperature measurements (in a short period of time) are not the same due to rounding errors of the Analog to Digital Converter (ADC) of the temperature sensor, electrical disturbances etc. Assume the mean of these random errors is close to zero, so an averaging algorithm will calculate a temperature which is close to the real temperature plus or minus the bias of the sensor. Please note that the averaging is done locally for each node. An average of K samples, per epoch, will be taken. From time stamp $\tau - K + 1$ up to and including time stamp τ. Where $\tau - K + 1 \ldots \tau$ are time stamps in one epoch. This averaged value is sent to the database, this value is $T_{(i,t)}$, where t is the time of the epoch.

$$T_{(i,t)} = \frac{1}{K} \sum_{k=1}^{K} T_{(i,\tau_k)} \tag{3}$$

Which value of K to choose? The probability that mean of the random error will be closer to zero if the value of K increases is higher. The value of K is restricted due to practical limitations, sampling of a sensor takes time and energy, so K can not be set too high. More sophisticated approaches exist, like a weighted average or a Kalman filter, but are undesirable due to their computational complexity and constraints. Recall that these computations have to be performed on the node itself. Sending all K measurements (of one epoch) to a central place where it can be processed, is not an option due to the relatively high energy costs of data communication in sensor networks.

5.3 Phase three, optimize system response

The third and final step is to optimize the system accuracy. Notice that the third step of the algorithm is performed centrally, using measurement data stored in the database. When trying to make a simple model of the temperature one can try to fit a line, a plane, or a higher order curve, between measured temperatures of different nodes at the same epoch, using least squares regression. The temperature for node i can be estimated with the following equation.

$$\hat{T}_i = \alpha_0 + \sum_{p=1}^{P} \alpha_p x_i^p + \sum_{q=P+1}^{P+Q} \alpha_q y_i^{q-P} \tag{4}$$

Where x_i and y_i are x and y position respectively of sensor node i and $\alpha_0 \ldots \alpha_Q$ coefficients to be determined. The plane which fits the best, has has the lowest sum of residuals S_r, equation 5.

$$S_r = \sum_{i=1}^{N} \left(T_i - \left(\alpha_0 + \sum_{p=1}^{P} \alpha_p x_i^p + \sum_{q=P+1}^{P+Q} \alpha_q y_i^{q-P} \right) \right)^2 \tag{5}$$

Lowest sum of residuals means that S_r has to be minimized with respect to zero. Minimize S_r means, differentiate S_r to each coefficient α_i, equation 6, and set the equations to zero.

$$\frac{\partial S_r}{\partial \alpha_i} = 0, \qquad i = 0, \ldots, Q \tag{6}$$

The coefficients $\alpha_0 \ldots \alpha_Q$ can be calculated. In matrix notation.

$$[[\mathbf{Z}]^{\mathbf{T}}[\mathbf{Z}]]\mathbf{A} = [\mathbf{Z}]^{\mathbf{T}}\mathbf{T} \tag{7}$$

$$\mathbf{A} = [[\mathbf{Z}]^{\mathbf{T}}[\mathbf{Z}]]^{-\mathbf{1}}[\mathbf{Z}]^{\mathbf{T}}\mathbf{T} \tag{8}$$

Where $\mathbf{A}$ is the vector of $\alpha_0 \ldots \alpha_Q$ and $\mathbf{T}$ the vector of measured temperatures T_i. Let $[\mathbf{Z}]$ be the following ma-

trix;

$$[\mathbf{Z}] = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^P & y_1 & y_1^2 & \dots & y_1^Q \\ 1 & x_2 & x_2^2 & \dots & x_2^P & y_2 & y_2^2 & \dots & y_2^Q \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^P & y_n & y_n^2 & \dots & y_n^Q \end{bmatrix} \tag{9}$$

5.3.1 Order of polynomial

The correction, of the bias, can be improved in two ways. First, do not use measurement data from one epoch, but calculate corrections for the biases for many epochs and average the result. A bad correction for a single sensor, e.g. due to noise in a single epoch is than slightly suppressed. Note that more samples do not necessary mean better results. When two estimated distributions of the temperature are more or less the same (e.g. measurements from consecutive epochs) the calculated correction values will also be the same. So averaging two more similar values makes no difference. When the temperature changes, different distributions of the temperature of the deployment scenario can be made. Due to the differences in temperature, different correction values will be calculated. A bad correction for a single sensor of one distribution (one epoch) will be out averaged by multiple good estimations. The more different distributions of the temperature exist, the better the result. This results in equation 10, where p is the order of the polynomial, $p = 1 \dots 4$ and E is the number of epochs, and i is the number of the sensor.

$$\lambda_{(i,p)} = \frac{1}{E} \sum_{e=1}^{E} \lambda_{(i,p)_e} \tag{10}$$

The second way to improve the correction is to average the correction calculated by different order of polynomials. In simulations it is noticed that, a bad correction for one sensor for one specific order of polynomial will be out averaged by averaging calculated corrections of all polynomials ($p = 1 \dots 4$). In some cases due to this averaging the correction is better than, a correction calculated by a single p^{th} order polynomial. Sometimes a single p^{th} order polynomial is better. The best choice can only be determined afterwards, and only in simulations when real values of the biases are available. Averaging the calculated corrections of different order polynomials will result in most cases in good estimations of the corrections. This is written out in equation 11, where $P = 4$.

$$\lambda_i = \frac{1}{P} \sum_{p=1}^{P} \lambda_{(i,p)} \tag{11}$$

5.3.2 Decision

The last step in the algorithm is to decide whether the calculated corrections should be applied or not. If very accurate sensors are used, the model of the distribution of the temperature might be not good enough. When applying the calculated corrections, the system accuracy might get worse. On the other side, the model of the distribution of the temperature might be good enough if very bad sensors are used. In that case the system accuracy improves. Recall that one has to look at the accuracy of the system as a whole. When the error $(\epsilon_i - \lambda_i)$ for 9 sensors is increased with $0.1\,^{\circ}\mathrm{C}$, and for one sensor the error is decreased with $1\,^{\circ}\mathrm{C}$, it is still an improvement of the system as a whole. So some sort of decision function must be made which decides to apply the corrections or not. As stated earlier the assumption has been made that the bias is Gaussian distributed, assume $\mu = 0$ and $\sigma = 1$, note[1]. Some properties of the Gaussian distribution are going to be used. The first question to be answered, what is the system accuracy before applying the algorithm. Recall equation 1, the system accuracy is the summation of the absolute value of all bias errors. When having N sensors (and N is large), the average value of the absolute value of the bias can be calculated with equation 12.

$$\bar{\epsilon} = \frac{1}{N} \sum_{i=1}^{N} \mid \epsilon_i \mid = 0.7978\sigma \qquad \epsilon_i \; is \; \mathbf{N}(0, \sigma^2) - dist. \tag{12}$$

Lets call this value the average absolute value of the bias, $\bar{\epsilon}$. So when $\sigma = 1$ and having 50 sensors, the system accuracy would be $50 * 0.7978 \approx 39.9^{\circ}\mathrm{C}$. Assume that the sensors are bought in a store. The chance that a random set of sensors have a better system accuracy is 50%. The chance that a random set of sensors have a worse system accuracy is also 50% (symmetric properties). When deciding to apply the calculated corrections, based on a function with $\bar{\epsilon}$ as parameter, the system accuracy might get worse when using a good set of sensors. So $\bar{\epsilon}$ have to be made a little bit lower to cope with a good set of sensors. Lets call this lowered boundary, $\bar{\epsilon}_{low}$. The value of $\bar{\epsilon}_{low}$ depends on a certain percentage p and the number of sensors N. Assume the boundary is lowered that much that when buying a random set of senors, the chance that the system accuracy of that set is above that boundary is p, e.g. $p = 95\%$.

In order to calculate the lower boundary, two equations from statistics. If the biases $\epsilon_1 \dots \epsilon_N$ are mutually independent, and all $\mathbf{N}(\mu, \sigma^2)$ distributed, than equation 13 and 14 can be used to calculated the sum and sample-mean respectively.

$$S_N = \sum_{i=1}^{N} \epsilon_i \qquad \mathbf{N}(N\mu, N\sigma^2) - distributed \tag{13}$$

$$\bar{\epsilon}_N = \frac{1}{N} \sum_{i=1}^{N} \epsilon_i \qquad \mathbf{N}\left(\mu, \frac{\sigma^2}{N}\right) - distributed \tag{14}$$

[1] The value of σ can be derived from e.g. the data sheet for real sensors.

To calculate a boundary for e.g. $p > 95\%$, the cumulative table of a standard normal distribution has to be used. In this table, one can find that, $p > 95\% = \Phi(1.65)$. Equation 15 is used to show the calculation of the boundary, where S_N^* is the standardized value of S_N.

$$P\left(S_N \leq \bar{\epsilon}_{low} N\right) = P\left(S_N^* \leq \frac{\bar{\epsilon}_{low} N - \bar{\epsilon} N}{\sqrt{\frac{\sigma^2}{N}}\sqrt{N}}\right) = \Phi(1.65) \tag{15}$$

This value, $\bar{\epsilon}_{low}$ will be used in an equation to decide to apply the calculated corrections or not. Note that, the higher value of p, the lower the boundary. Note also, for a large number of nodes, this boundary is not lowered that much.

The last step is the real decision of the decision function. Please notice that the empirical decision function is found during experiments. At this stage of the algorithm the estimations of the biases, all λ_i are known. In the ideal case all λ_i are equal to ϵ_i. Unfortunately the curve fitting does not fit every real physical temperature T_i^*. So at every sensor i a small difference between the estimated and the real bias remains. Lets call this error at sensor i, η_i. The next equation is related to the sum of residuals (equation 5), but due to all the averaging which have been carried out there is no single physical temperature and no single estimated temperature anymore. When summating the square of all λ_i more or less the same results is obtained although. As stated before there is an error η_i at every sensor i, so the λ_i's in the next equation can be replaced with ϵ_i and η_i, which are both unknown. Equation 16.

$$\sum_{i=1}^{N} \lambda_i^2 = \sum_{i=1}^{N} (\epsilon_i + \eta_i)^2 \tag{16}$$

When perfect sensors are used, all ϵ_i are zero, so the only variable left is the amount of error at each sensor squared, η_i^2. This summation of η_i^2 is the best system accuracy which is possible. This value seemed to be important in the decision function. Perfect sensors are impossible, all sensors have a bias error ϵ, so the amount of error (η_i^2) at each sensor must be estimated. One knows that on average of the absolute value of all ϵ_i's, this value is $\bar{\epsilon}$ as mentioned earlier. So the value of the summation of all η_i^2 can be estimated, this is done in equation 17, note[2]. Recall that $\bar{\epsilon}$ is a function of σ.

$$\sum_{i=1}^{N} \eta_i^2 \approx \sum_{i=1}^{N} \lambda_i^2 - N\bar{\epsilon}^2 \tag{17}$$

The final, empirical found decision function becomes equation 18. When the left side of the equation is larger than the right side of the equation, the algorithm should not be applied.

$$(\sum_{i=1}^{N} \lambda_i^2 - N\bar{\epsilon}^2) - (N\bar{\epsilon}^2)\bar{\epsilon}_{low}^2 \geq N\bar{\epsilon}_{low}^2 \tag{18}$$

During simulations it is noticed that the left side of the previous equation results in more or less a straight line when simulating the proposed calibration algorithm for different values of σ. For small values of σ the left side of the equation is larger and therefore the accuracy of the system as a whole decreases when applying the corrections. When the value of σ increases, the line of the left side of the equation intersects line of $N\bar{\epsilon}^2$ at a certain point, note[3]. For this and larger values of σ the calculated corrections of the biases could be applied. Note that $\bar{\epsilon}_{low}$ has been introduced to raise the left side of equation 18 and to lower the right side of this equation a little bit for a higher value of percentage p. Recall that a higher percentage of p, e.g. $p = 95\%$ results in a lower value of $\bar{\epsilon}_{low}$

6 Simulations

When performing a simulation run, figure 1 can be made. The standard deviation of the bias is simulated from 0 to 0.5 (depicted on the x axis). On the y axis the average amount of error per sensor in $^\circ$C (absolute value) is depicted. In the graph itself several system accuracies are depicted. The solid line is the original system accuracy. System accuracies are depicted when applying corrections by single order of polynomials (1^{st} $\ldots 4^{th} order$). The system accuracy is also depicted when the corrections calculated by the several single p^{th} order polynomials are averaged, and finally the system accuracy of these averaged corrections are depicted including the decision function which decides to apply the algorithm or not.

Many simulation runs have been carried out and many could be placed in this paper. Two simulations (office environment) have been chosen to show the different results of the different order of polynomials. The different results when averaging the calculated corrections and the decision function. The decision in one simulation is very accurate, in the other simulation it is very conservative. The results of the first simulation run are depicted in figure 1. The performance of the algorithm and the correctness of the decision function can be measured. This can be done by calculating the amount of 'improvement', 'missed' and 'degradation' per node (per standard deviation) in degrees Celsius. In the first simulation example, the decision function decides to apply the calculated corrections when σ is 0.175. The system accuracy gets worse, although $6.4m^\circ$C per node is not worth mentioning it. For values of σ is 0.2 and higher the system accuracy improves (the summation gets lower).

[2] Note that in theory this value can be negative due to this calculation, this is no problem because it is used as a sort of threshold.

[3] Recall that $\bar{\epsilon}$ is a function of σ

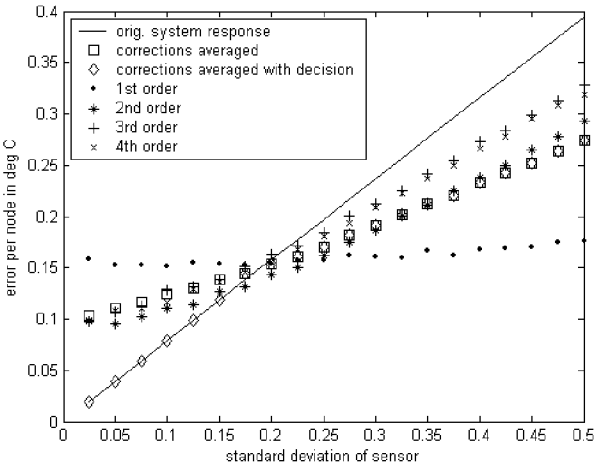

Figure 1. Simulation 1 office environment, 20 nodes

When running the same scenario a second time, new biases are generated and the nodes are placed randomly again (thus the nodes are placed at other places), the results of the algorithm will therefore also be different. In the second simulation run, a relative bad set of sensors have been simulated. A set of sensors have an absolute average bias of 0.7978σ. In this set the absolute average value of the bias is $\approx 0.9\sigma$. Due to this high value, the algorithm decided to apply the calculated corrections too late. It could apply the corrections for biases with a standard deviation of 0.15, but it decided to apply the corrections for a standard deviation of 0.425 and higher, as can be seen in figure 2. The amount 'missed' is not

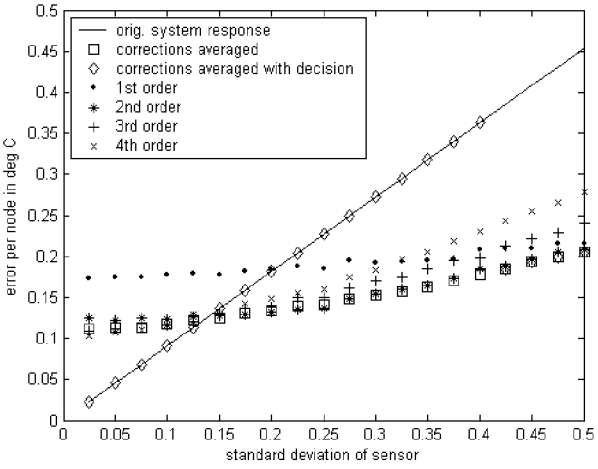

Figure 2. Simulation 2 office environment, 20 nodes

that much per node in the beginning, when $\sigma = 0.15 \ldots 0.25$. Improvements are getting noticeable for standard deviations of the bias of 0.275 and higher, $0.1\,^{\circ}\mathrm{C}$ per node.

7 Results

Data was received from 24 distinct nodes. Temperature readings of the LM92 temperature sensors [5] seemed to be valid, all between $23.125\,^{\circ}\mathrm{C}$ and $26.125\,^{\circ}\mathrm{C}$. To calculate the correction of the biases of the sensors the algorithm has been applied on a section of the measurement data. Only epochs where more than N (e.g. $N = 8$) nodes delivered temperature readings were used. The tests were influenced with communication problems, epochs were more than 20 nodes delivered measurements were rare. Increasing N did not make that much difference. In total 185 epochs were used. Calculations for the corrections of the biases (for every node) were made for 1^{st}, 2^{nd}, 3^{rd} and 4^{th} order polynomials. These calculated corrections were averaged as described before. The algorithm decided not to apply the corrections of the biases. Calculated corrections were too high although close to the boundary (for different values of p) were it could apply the corrections. The average absolute value of the bias when having a large amount of LM92 temperature sensors is $0.08776\,^{\circ}\mathrm{C}$. Note that the resolution of the sensor is $0.0625\,^{\circ}\mathrm{C}$ so the algorithm should correct the measurements in the order of 1 or 2 times the resolution which is almost impossible in an uncontrolled environment. When looking at the results of the algorithm the calculated corrections seemed quite reasonable. From the 24 calculated corrections, 6 calculated corrections had a higher absolute value than $0.08776^{\circ}\mathrm{C}$, but these higher values have probably too much weight, so the algorithm decided not to apply the corrections. The highest calculated correction was $0.36\,^{\circ}\mathrm{C}$. This value is a little bit more than the maximum bias error ($0.33\,^{\circ}\mathrm{C}$) according to the data sheet of the LM92.

According to the data sheet and assuming that the readings of the LM92 are correct, temperature readings of the internal temperature sensor of the MSP430F149 micro processor [3] of the sensor node should be between $10\,^{\circ}\mathrm{C}$ and $40\,^{\circ}\mathrm{C}$. Looking at the results, temperatures between $17.87\,^{\circ}\mathrm{C}$ and $32.49\,^{\circ}\mathrm{C}$ were measured, so the measurements seemed to be valid. Temperature readings of sensor node 1 are depicted in figure 3. The internal temperature reads always $\approx 6.5\,^{\circ}\mathrm{C}$ higher than the LM92 temperature sensor, the difference between both bias errors. The temperature course is more or less equal. The same phenomena is observed at all other nodes.

The calibration algorithm is also applied on measurement data generated by the internal temperature sensor. The algorithm decided to apply the corrections (for different values of p). In table 1 the calculated corrections for the several order of polynomials are depicted, as well

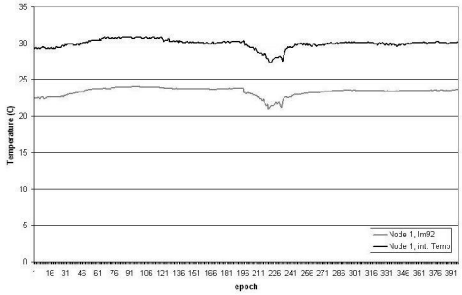

Figure 3. Temperature readings of node 1

as the average correction (of the several different orders of polynomials). In the last column the pairwise (relative) calibration relationships between the LM92 and the internal temperature sensor are depicted. The pairwise calibration relationships are calculated with equation 19, where E is the number of epochs. If there is no temperature measurement for an epoch available the difference for that epoch is set to zero.

node	1^{st}	2^{nd}	3^{rd}	4^{th}	aver.	pair
1	-3.14	-2.15	-0.10	0.42	-1.24	-6.75
2	0.06	0.38	0.06	-0.02	0.12	1.43
3	-4.96	-4.97	-4.25	-3.56	-4.43	-4.46
4	-2.95	-2.81	-0.97	-0.43	-1.79	-3.56
5	2.07	2.45	1.54	1.08	1.79	5.44
6	-0.56	-1.02	-1.46	-0.49	-0.88	-1.95
7	-0.35	0.15	-0.11	-0.67	-0.25	2.25
8	1.48	2.14	0.25	0.09	0.99	-1.77
9	0.13	-0.23	0.60	0.08	0.14	1.16
10	-2.99	-1.78	0.08	0.20	-1.12	-2.05
11	1.61	1.56	0.44	-0.48	0.78	-1.08
12	4.38	1.79	1.44	1.17	2.19	5.63
13	0.07	0.13	0.08	0.05	0.08	-0.30
14	-1.93	-2.01	-0.98	-0.80	-1.43	-6.83
15	6.11	5.75	3.77	3.36	4.75	4.76
16	0.42	0.19	-0.08	-0.65	-0.03	-3.71
17	1.22	0.87	0.44	0.49	0.76	0.55
19	0.25	0.52	0.70	0.54	0.50	-0.20
20	-4.44	-3.71	-2.64	-1.25	-3.01	-6.95
21	1.81	1.73	0.43	0.27	1.06	1.78
22	-3.59	-3.24	-1.02	-1.67	-2.38	-7.89
23	-4.25	-3.48	-2.00	-0.57	-2.57	-0.71
24	-2.15	-1.01	0.16	0.01	-0.75	-4.14
25	-0.85	-1.09	-0.43	0.03	-0.58	-4.73

Table 1. results of int. temperature sensor

$$\frac{1}{E}\sum_{e=1}^{E}(T_{e_{LM92}} - T_{e_{int}}) \tag{19}$$

During simulations and testing, the assumption was that for both sensors the biases are Gaussian distributed, with mean $\mu = 0$. Due to the fact that no temperature sensor is used which is calibrated to an absolute reference temperature, the real value of μ for both temperature sensors can not be determined. The μ of the biases of LM92 temperature sensor as well as the internal temperature sensor will probably be close to zero, according to measured temperatures and priori knowledge. But if $\mu = +0.2\,^{\circ}\mathrm{C}$ or $-0.05\,^{\circ}\mathrm{C}$, one can not say. When averaging all temperature measurements of the LM92 temperature sensors, the average temperature is $24.28\,^{\circ}\mathrm{C}$. The average temperature of all the measurements of all internal temperature sensors of the MSP430 is $24.64\,^{\circ}\mathrm{C}$. When the average temperatures are equal, one can conclude that the value of μ for both sensors is equal, and probably close to zero. Although the difference in average temperature is not that big, the values of μ are definitely not the same. Therefore one can not deduce the real value of μ for the LM92 and the internal temperature sensor. When assumed that the temperature readings of the LM92 are correct a pairwise calibration relationship between the LM92 and the internal temperature sensor can be made. This pairwise calibration relationship is an estimation of the bias of the internal temperature sensor. When comparing these values with the averaged calculated corrections, these values must be in the same order. Looking at the results, the average correction of node 8 and 11 are different in sign with respect to their pairwise calibration. Looking in table 1 and 2 one can see that node 8 and node 11 are placed close to node 14, 15 and 16. These nodes have relative high (absolute value) bias errors. Due to the fact that a curve has to be fit through the measured temperatures, the algorithm calculates corrections for node 8 and 11, which are affected by biased measurements of nodes 14, 15 and 16. Therefore the calculated corrections for node 8 and 11 are not correct. Looking at the pairwise calibration relationships, and assuming these relationships are more or less correct, node 1, 5, 12, 14, 20 and 22 have relative large bias errors. Using only a 1^{st} order of polynomial (a plane) to calculate the corrections would produce better results. The problem is that this conclusion is based on measurements of a temperature sensor which measures at the same position and time the temperature. In an application where no reference temperature sensor is placed this conclusion can not be drawn. Again, assume the temperature readings of the LM92 temperature sensors are correct.

node	pos. X	pos. Y	node	pos. X	pos. Y
1	0.5	1	14	5.5	1.5
2	0.5	2.5	15	3.5	3
3	1.5	1	16	3.5	2
4	3	0.5	17	4	3
5	1.5	0.5	18	4.5	2
6	5.5	1	19	6	1
7	2	0.5	20	3	2
8	3.5	2.5	21	1	3
9	0.5	3	22	6	1.5
10	0.5	0.5	23	1	0.5
11	4	2	24	2.5	2
12	1	1	25	4.5	3
13	2	1			

Table 2. Position of the nodes

8 Future work

During the experiments to test the calibration algorithm, sensor nodes were placed on or close to the ground. This means that the algorithm is tested in a 2 dimensional scenario. In a real life scenario it is likely that sensor nodes are placed on different heights (meters difference). The calibration algorithm should be adapted so that it can be used in a 3 dimensional environment. This should be

simulated and tested. The calibration algorithm has been tested in a relative small room. Temperature differences were therefore relative small. When a sensor network is applied in a larger area, it is likely that the difference in temperature increase and that there is less correlation in temperature between groups of nodes that are physically far away from each other. In this case the area should be separated into smaller areas, so the algorithm becomes a more localized calibration algorithm. Separation can be done by dividing the area in smaller areas on beforehand. More sophisticated approaches can be thought of, like a separation based on temperature readings. Temperature sensors which are measuring the same temperature course in time can be grouped together.

A final question should be answered. What to do with sensor readings that are probably measurements errors. Some measurement values can be filtered out very easily. Due to errors, e.g. a communication error with the sensor or a conversion error on the sensor, the sensor probably reads the minimum or maximum sensor reading which is possible. Assume measurements are not performed at the boundaries of the sensor, so these extreme sensor readings can be detected and removed quite easily. When one or two sensors are measuring slightly different temperatures, it is difficult to state if the measurements are correct or not. The office scenario where the algorithm is tested is taken as example. Assume all sensors are measuring between $22\,^{\circ}\mathrm{C}$ and $25\,^{\circ}\mathrm{C}$, one sensor measures between $29\,^{\circ}\mathrm{C}$ and $31\,^{\circ}\mathrm{C}$. What to with these values, it is a little bit high but not impossible in an office. Making a model of the distribution of the temperature with or without these high temperature readings makes a difference. Therefore the algorithm might make a wrong decision in order to apply the algorithm or not. How to filter measured data in order to improve the results of the calibration algorithm should be researched further. What to do if the decision function decides that the calculated corrections must be applied. Assume all corrections are less then the maximum expected (absolute) bias error except one. Should one apply the calculate correction for this sensor, or the maximum expected (absolute) bias error or should this calculated value (or all values) be neglected. These questions can be researched in the future. Some additional tests can be made to check the calculated corrections. The calculated corrections should follow a Gaussian distribution. If e.g. the μ is not close to zero or the shape does not look like a Gaussian distribution the calculated corrections are unlikely. Additional tests can be researched further.

A model of the distribution of the temperature can be made to check for measurement errors. The model can also be used to inter- or extrapolate measurement values. Useful when measurement data is missing, or to check the validity of measurement data. A small amount of accurate sensors can be used to calibrate a large amount of less accurate sensors. These possibilities should be explored.

9 Conclusion

In order to improve the accuracy of sensor measurements, less accurate sensors must be calibrated. Without any question the best way to do this is, is manually. However, the labor needed for manual calibration makes this approach unusable. The proposed calibration algorithm is suitable for calibration of sensors in sensor networks. Simulations and practical tests have been carried out with temperature sensors. These simulations and tests showed that the accuracy of the temperature sensors can be improved. The amount of improvement depends on the environment, in a clean environment, where the distribution of temperature is smooth, large improvements of the accuracy are possible. Due to the conservative behavior of the algorithm, the chance of a decrease of accuracy (of the whole system) is small. Therefore the proposed calibration algorithm can be used to improve the accuracy of the measurements, without any harm. The proposed algorithm is an off-line algorithm, so it requires no additional hard- or software on the sensor nodes. Besides the possibility to calibrate sensors, the model of the distribution of the temperature can be used to check measurement values for validity. It is possible to inter- or extrapolate missing measurement data. This can be exploited while estimating (calculating) a measurement is more efficient in terms of energy than sampling a real physical sensor. This is useful in a WSN.

References

[1] V. Bychkovskiy. Distributed in-place calibration in sensor networks. Master's thesis, University of California at Los Angeles, June 2003.

[2] V. Byckovskiy, S. Megerian, D. Estrin, and M. Potkonjak. A collaborative approach to in-place sensor calibration. In *Proceedings of the Second International Workshop on Information Processing in Sensor Networks (IPSN)*, volume 2634 of *Lecture Notes in Computer Science*, pages 301–316. Springer–Verlag Berlin Heidelberg, 2003.

[3] T. Instruments. MSP430F149 Data sheet, http://focus.ti.com/lit/ds/symlink/msp430f149.pdf.

[4] A. LaMarca, W. Brunette, D. Koizumi, M. Lease, S. B. Sigurdsson, K. Sikorski, D. Fox, and G. Borriello. Making sensor networks practical with robots. In *Pervasive '02: Proceedings of the First International Conference on Pervasive Computing*, pages 152–166. Springer-Verlag, 2002.

[5] N. Semiconductor. Temperature sensor LM92, http://www.national.com/ds.cgi/LM/LM92.pdf.

[6] K. Whitehouse and D. Culler. Calibration as parameter estimation in sensor networks, 2002.