# Feasibility of shifting decision support tools for quality estimation in food logistics to the sensor node level

Alexander Dannies[1], Javier Palafox-Albarrán[1], Reiner Jedermann[1]and Walter Lang[1]

[1]Institute for Microsensors, -actuators and -systems, University of Bremen, Germany

**The topic of this paper is the verification of the feasibility of execution of quality estimation algorithms in food logistics (cold-chain transportation) on the sensor node level. In the Intelligent Container Project we want to achieve monitoring of environmental parameters to estimate the current and future quality state of the perishable goods. Normally, a wireless sensor network is only used for data collection and transmission to a gateway device which acts as a bridge between the local environment and the end-user or logistics control center. It can either process the raw data or send them directly over a cellular network or satellite link. Data processing at node level has several benefits such as reduction of the amount of transmitted data to the gateway, thereby reducing the energy consumption on the nodes. In this paper we verify the feasibility of shifting decision support tools to the sensor node level. The conclusions show that it is possible to locally process the temperature and thus estimate the quality by running algorithms implemented in Java directly on wireless sensor nodes.**

**On-node data processing, Java, Wireless sensor nodes**

## I. INTRODUCTION

QUALITY ESTIMATION of perishable goods in the cold-chain of food logistics can be used to obtain benefits both on the retailer and the consumer side. Our goal in the Intelligent Container Project [1] is to achieve improvement of the traceability of foods and realization of autonomous monitoring of the food quality during transport. By making use of this additional information in the logistics a better performance of the transport planning can be achieved, which leads to reduction of food losses, prevention of unnecessary transport of goods, reducing $CO_2$ emissions and higher quality for the consumers.

To make this become reality a wireless sensor network (WSN) is used to collect environmental data inside a refrigerated container. The sensed parameters include temperature, humidity, acceleration or gas concentrations ($CO_2$/$C_2H_4$). A gateway device with extended communications capabilities (satellite link) bridges the gap between the local WSN and the remote logistics control center. Normally, the sensor nodes only act as collectors and send the data periodically to the gateway device where the whole data of all sensor nodes of the network are gathered. Decision making processes can be made on the gateway device or, if all data is sent to a remote location, processed there. The question arises if it is possible to shift the decision making to the sensor node layer. Local processing will bring various advantages such as limiting high amount of data transmission, reducing high costs for satellite communications and preventing eventual data loss due to humidity or obstacles inside the container. The aim of our paper is to verify whether it is feasible to preprocess data directly on the device on which it is collected.

In section 2 we present several example algorithms used for decision making in our specific field of application. Section 3 presents two different hardware platforms that we chose to measure the performance of complex Java algorithms on hardware- constrained platforms with limited processing capabilities. Section 4 summarizes the test results of the algorithms on the platforms. Finally, we conclude in section 5.

## II. ALGORITHMS FOR QUALITY ESTIMATION

### A. *Temperature prediction in fruit logistics*

For estimating the quality of perishable goods in fruit logistics we have to take into account that fruits, in our case the bananas, are subject to metabolic processes which are dynamic. Bananas are “breathing”: they ingest oxygen and exhale $CO_2$ and the plant hormone ethylene ($C_2H_4$). Ethylene is also ingested by surrounding bananas inducing or accelerating the ripening process. Bananas are harvested green and transported in hibernation when the surrounding temperature is below 13 °C. Therefore, measuring the temperature during transportation can give clues to determine if the ripening process has started and how the remaining shelf life would change during the rest of the course.

The Feedback-Hammerstein model (Figure 1) is used to represent the factors affecting the temperature inside refrigerated containers transporting perishable goods.

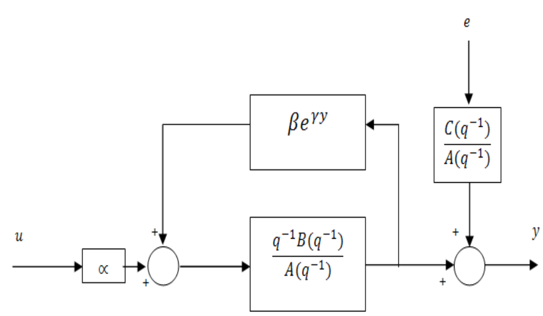


**Figure 1: Model of the Feedback-Hammerstein-algorithm**

A naive representation of the factors can be done by a SISO (single input single output) linear dynamic system in which the input is the air supply and the output is the spatial point of interest. However, in reality it is only a simple model of the main contributor to the temperature pattern. Several other factors affect the speed of the cooling down.

To improve the accuracy of the model, other contributors are considered as well: first is the heat, produced by respiration of living goods such as fruits and vegetables; second is the thermal loss, affecting the correct cooling of the good.

The so-called Feedback-Hammerstein model is using a static pseudo-linear feedback system for taking the effect of organic heat into account.

Details of the algorithm are described in [2].

### *B. Temperature prediction in meat logistics*

In the case of meat, the cargo does not produce organic heat and the linear feedback block of the model is removed, the system is considered to be linear.

Initially, it can be thought that systems of high linear order would improve the accuracy of a linear model. Generally speaking, for complex linear systems, this is true. It is important to notice the decaying behavior of the output data due to the refrigeration process.

In considering a linear system, an exponential discrete time decaying system of a refrigerated chamber or container can be simply described as one of the order of one with its unique pole. The closer the pole to 1, the higher the delay of the system. A second pole can give more precision. However, orders higher than two are not considered due to the high possibility of computation of complex conjugate poles that would characterize oscillations in the modeled system.

### *C. Parameter adaptation algorithm*

In order to provide an accurate system, the model parameters of the FH and linear system have to be estimated by a parameter adaptation algorithm.

Its principal advantage is that the conventional recursive matrix-based linear system identification algorithms as those presented in [6] can be applied to estimate the parameter matrix. The recursive form of this algorithm is given by Equation 1. Where $\varphi(t)$ and $\Theta(t)$ are the so-called observation and parameter vector correspondingly. The arrangement of the elements depends on the considered model, as shown in table 1. The prediction error is described in Equation 2, $P(t+1)$ is an adaptation matrix to perform the minimization of using Recursive Least Squares method, and is the observation matrix that contains the input and the output data. As can be noted not any matrix inversion is needed in the algorithm, which makes it very energy efficient.

$$\Theta(t+1)=\Theta(t)+\left(P(t+1)\varphi(t)\right)^T\varepsilon(t) \quad (1)$$

$$\varepsilon(t) = y(t) - \Theta(t)^T \phi(t-1) \quad (2)$$

$$P(t+1) = \frac{P(t) - P(t)\varphi\varphi^T\left(\frac{P(t)}{\varphi^T P\varphi + \lambda(t+1)}\right)}{\lambda(t+1)} \quad (3)$$

$$\lambda(t+1) = \lambda_o \cdot \lambda(t) + 1 - \lambda_o \quad (4)$$

**Table 1: Arrangement of the elements in the algorithm matrices**

| System | Symbol | Arrangement of the elements into the matrices |
|---|---|---|
| Feedback Hammerstein | $\varphi(t)$ | $[-y(t), u(t-1), (e^{\gamma y(t)} - \gamma y(t))]$ |
| | $\Theta^T(t)$ | $[a_1^*, b_1\alpha, \beta b_1]$ |
| Linear Order 2 | $\varphi(t)$ | $[-y(t), -y(t-1), u(t-1)]$ |
| | $\Theta^T(t)$ | $[a_1, a_2, b_1]$ |
| Linear Order 1 | $\varphi(t)$ | $[-y(t), u(t-1)]$ |
| | $\Theta^T(t)$ | $[a_1, b_1]$ |

### *D. Quality change in fruit logistics*

In our case the banana serves as an exemplary fruit in fruit logistics to illustrate quality change. The banana has different ripening states: green life, climacteric and ripening which are reflected in their color from green (harvesting) to yellow (point in time with best look and flavor). The beginning of decomposition with brownish coloring is undesirable and equals to a low quality. To ensure that the quality of the bananas during transport is within specified limits, the temperature can be used to calculate the remaining shelf life.

### *E. Quality change in meat logistics*

Usually a distinction is made between the quality and the freshness of a product. In the context of this paper we use the term quality in the sense of freshness of the product.

An indicator for the quality of meat is the presence of bacteria on the sample. An unacceptable level of the bacterial accumulation is accompanied by discoloration, changes in texture as well as a specific smell and flavor.

The majority of the models to determine the speed of bacterial growth depend on temperature, because it is the main factor influencing it. The so-called combined model, describe the bacterial growth as a function of time as well. As the name suggests, it is a combination of the Gompertz- and Arrhenius-model; details are described in [7].

The Arrhenius equation (5) gives a description of the relation between the speed of chemical reactions and temperature.

$$k = A \cdot e^{-\frac{E_a}{RT}} \quad (5)$$

The reaction constant $k$ can be calculated using the pre-exponential factor $A$, the activation energy $E_a$, the universal gas constant $R$ and the temperature $T$.

A Gompertz function (equation 6) is a type of mathematical model for a time series, with slowest growth at the start and end of a time period. The function converges much slower to the future value asymptote than to the lower valued asymptote.

$$y(t) = a \cdot e^{b \cdot e^{c \cdot t}} \quad (6)$$

In equation 6 $a$ is the upper asymptote, $b$ sets the x displacement and $c$ sets the growth rate.

In the model, the variables that are independent of the temperature are set as constants, whereas those depending on it are calculated accordingly to the Arrhenius equation. If the temperature remains constant, only few mathematical operations have to be executed for each measurement interval. After temperature changes, all parameters of the Gompertz equation have to be re-calculated, which required more operations. The reversal point (inflection point of the curve) $xc_i$ has to be adjusted to the new temperature, depending on the current microbial counts $N_0$ according to equation 7. The bacteria content is calculated according to equation 8.

$$xc_i = \frac{ln\left(-ln\left(\frac{N_{ti-1} - N_0}{a}\right)\right)}{k_{T_i}} + t \quad (7)$$

$$N(t_i) = N_0 + a \cdot e^{-e^{k_{T_i}(t - xc_i)}} \quad (8)$$

$N(t)$ represents the germ concentration at the time of $t$
$N_0$ is the initial germ concentration
$a$ is a temperature-independent variable
$k_{T_i}$ is a temperature-dependent variable

## III. Hardware Platforms Sensor Nodes

We chose two different sensor node platforms for our tests. The Oracle SunSPOT [3] and Virtenio Preon32 [4] are described below. Our selection was based on the fact that these two platforms provide a virtual machine to execute Java code. The high-level programming language Java enables platform independent object oriented programming.

### A. Oracle SunSPOT

The sensor node “Sun Small Programmable Object Technology” (SunSPOT) from Oracle depicted in Figure 2 has a modular setup with three different layers: processor board, sensor board, energy pack in form of a 3.6V Li-Ion Battery which can be charged via USB.
We are using it in its 8th revision, which has the following specifications. As a CPU an ARM-architecture AT91SAM9G20 with a clock rate of 400 MHz is used. In addition to 1 MB of RAM it has a Flash memory of 8 MB available.

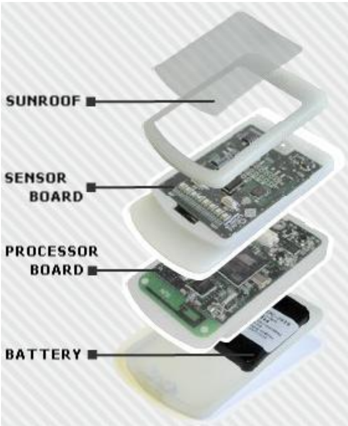


**Figure 2: Oracle SunSPOT sensor node [8]**

The Java Virtual Machine (JVM) used on this platform is Squawk [5] and runs without an operating system. Software modules running on the platform are so-called MIDlets, which are packed in a suite and deployed to the device.

### B. Virtenio Preon32

The Preon32 from Virtenio depicted in Figure 3 is also a modular system. The shuttle component can be connected to a sensor board with several available sensors like temperature, humidity, acceleration, light and many more.

As a CPU a Cortex-M3 ARM-CPU is used with a CPU rate up to 72 MHz. The working memory has a size of 64 kByte. It is extended by a flash memory of 256 kB of size.

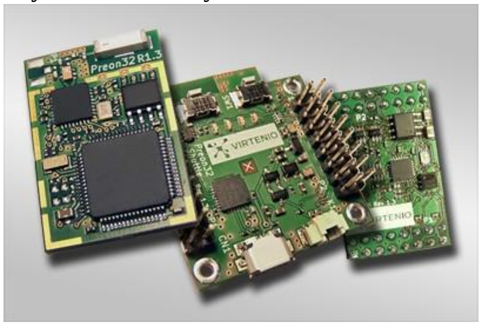


**Figure 3: Virtenio Preon32 sensor node [9]**

Virtenio has developed their own JVM which is similar to Squawk and also runs on bare metal. Programs are packed in so-called Proglets which are deployed on the device.

## IV. Test results

We used the algorithms for the quality estimation of fruit and meat respectively described in section II on the sensor nodes described in section III. The following figures show the results.

We tested the FH-algorithms for a data volume matching occurring within four days at a measurement interval of one hour, equivalent to 96 cycles.

The computation times for the different versions of the Feedback-Hammerstein algorithm executed on Preon32 is displayed in Figure 4.

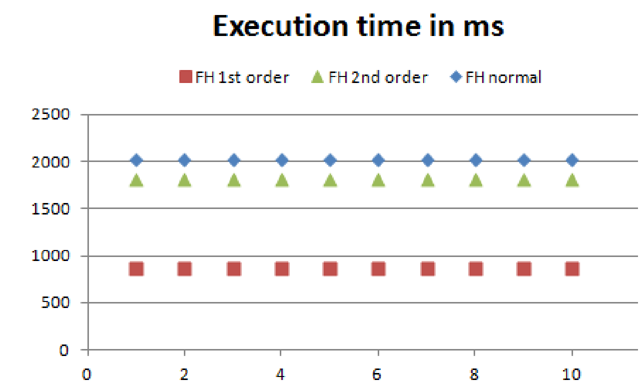


**Figure 4: Test results for different algorithms on Preon32**

The 1st order calculation is the fastest with approximately 868 ms. Using the 2nd-order approach the calculation time just about doubles and results in approximately 1815 ms. The slowest result is the normal FH-execution, which is ca. 17% slower and takes approximately 2022 ms.

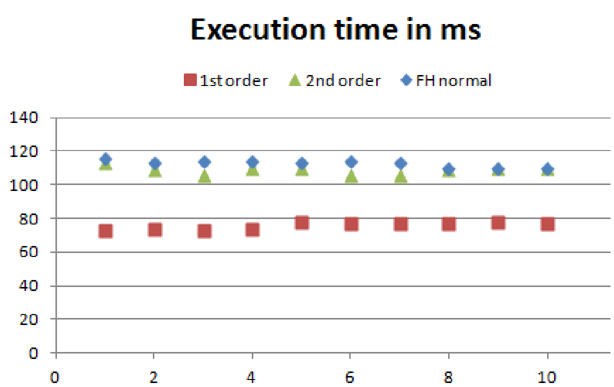


**Figure 5: Test results for different algorithms on SunSPOT**

The computation times for different versions of the Feedback-Hammerstein algorithm executed on the SunSPOT is displayed in Figure 5. The 1st-order calculation is the fastest with approximately 76 ms. Using the 2nd-order approach the calculation time just about doubles and results in approximately 109 ms. The slowest result is the normal FH-execution, which is ca. 3.5% slower and takes approximately 113 ms.

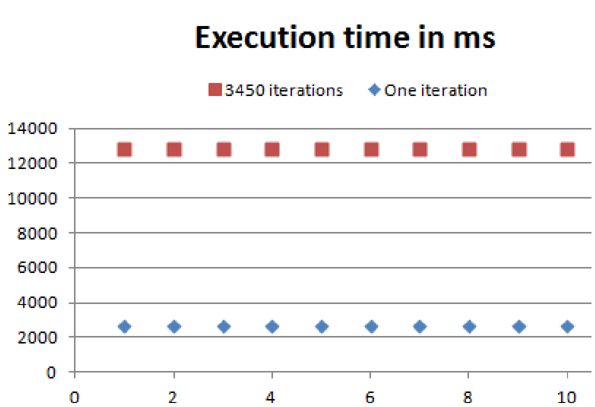


**Figure 6: Test results for combined model on Preon32**

Figure 6 shows the durations for the execution of the combined model algorithm for meat on Preon32. The figure gives the accumulated CPU time to evaluate the measured temperature of 346 intervals. We tested the algorithm with the following two extreme cases: Constant temperature over the full test duration and changing temperature for every iteration step. The mean values of calculation time per step were 2346 ms and 12817 ms, respectively. In the worst case, 13 seconds are required for calculation of the Gompertz model. But, assumed that the temperature persists at an almost constant value over several measurement intervals, only 2.5 seconds will be required for most intervals.

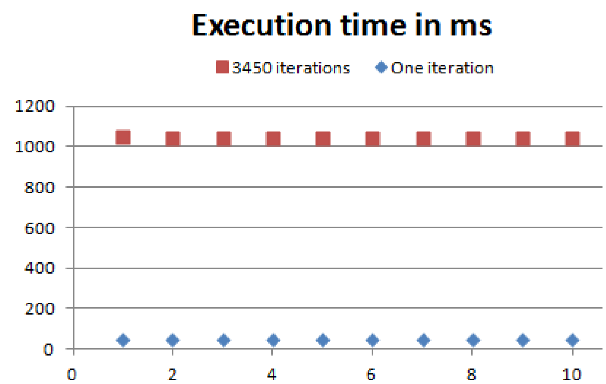


**Figure 7: Test results for combined model on SunSPOT**

Figure 7 shows the required time for the execution of the combined model algorithm with different numbers of iteration steps on the SunSPOT. The execution of the algorithm is much faster on this platform with mean values of 41 ms and 1042 ms, respectively. Preon32 requires about 20 times more CPU time for the execution of mathematical algorithms, but its performance is still sufficient for automated supervision tasks in food logistics.

## V. CONCLUSION

This paper has shown that it is feasible to run algorithms directly on the sensor node level instead of just collecting data and sending to a remote location for calculations.

We proved for two different hardware platforms that it is feasible to run complex Java algorithms directly on sensor nodes.

Local processing of sensor data will lead to reduced communication costs and increased system robustness.

Detailed information about product quality changes and temperature prediction will enable optimized planning of transport processes, thereby reducing costs, $CO_2$ and waste.

## ACKNOWLEDGMENT

The research project "The Intelligent Container" is supported by the Federal Ministry of Education and Research, Germany, under reference number 01IA10001.

## REFERENCES

[1] IMSAS . (2011). „The intelligent container: Networked intelligent objects in logistics“. [Online]. Retrieved May 14, 2012, from http://www.intelligentcontainer.com

[2] Palafox-Albarrán, J., Jedermann, R. & Lang, W. (2011). Energy-efficient parameter adaptation and prediction algorithms for the estimation of temperature development inside a food container. In: Cetto, A. J., Ferrier, J.-L. & Filipe, J. (eds.) *Lecture Notes in Electrical Engineering - Informatics in Control, Automation and Robotics*. Berlin: Springer.

[3] Oracle. (2011b). *SunSPOT* [Online]. Retrieved May 14, 2012, from http://www.sunspotworld.com

[4] Virtenio. (2011). 2.4 GHz Funkmodul „Preon32” mit überlegener Technik [Online] Retrieved May 14, 2012, from http://www.virtenio.com/de/produkte/hardware/preon32.html.

[5] Oracle. (2011a). *Squawk Development Wiki* [Online]. Retrieved May 14, 2012, from http://java.net/projects/squawk/pages/SquawkDevelopment

[6] Landau, I.D., Zito G.(2006). *Digital Control Systems: Design, identification and implementation*. Springer. London, 1st edition.

[7] Determination of the shelf life of sliced cooked ham based on the growth of lactic acid bacteria in different steps of the chain J. Kreyenschmidt, A. Hubner, E. Beierle, L. Chonsch, A. Scherer and B.Petersen Faculty of Agriculture, Institute of Animal Science, University of Bonn,Bonn,Germany

[8] http://www.sunspotworld.com/S314730_Sun_SPOTs_Web_Of_Things/images/spot-anatomy.jpg

[9] http://www.virtenio.com/de/assets/images/produkte/produkt_preon_serie.jpg