DOI 10.5162/sensor2017/A8.2

Temperature Estimation of Induction Machines Based on Wireless Sensor Networks

Y. Huang[1], C. Gühmann[2]
[1,2] TU Berlin, Chair of Electronic Measurement and Diagnostic Technology, Sekr. EN13, Einsteinufer 17, D-10589 Berlin, Germany
[1]yi.huang@campus.tu-berlin.de, [2]clemens.guehmann@tu-berlin.de

Abstract
The 4th-order Kalman filter (KF) algorithm is implemented in the wireless sensor node to estimate the temperatures of the stator winding, the rotor cage and the stator core in the induction machine. Three separate wireless sensor nodes are used to acquire and preprocess the input signals. The hall sensors are used to acquire the three-phase stator currents and voltages of the induction machine. All of them are processed to Root-Mean-Square (RMS) in ampere and volt. A rotary encoder is mounted for the rotor speed and PT1000 is used for the temperature of the coolant air. The processed signals in the physical unit are transmitted wirelessly to the host wireless sensor node, where the KF is implemented with fixed-point arithmetic in Contiki OS. Compared to the floating-point implementation, the fixed-point implementation has the same estimation accuracy of 97% at only about one-fifth of computation time. Temperatures of the machine could be monitored by an App on a smart phone with internet.

Key words: temperature estimation, induction machine, wireless sensor networks, fixed-point, Contiki

1 Introduction

Electrical machines are widely used in the industry, especially the increasing interest in electric and hybrid electric vehicles. The thermal behavior of an induction machine largely determines the maximum lifetime, the ability of over-load and also the accuracy in a high-performance controller [1]. Meanwhile the wireless sensor networks (WSN) have many applications such as industry, environment monitoring, tracking of things and internet of things. A number of methods for temperature monitoring of induction machines can be found in literature. Some of the methods do not provide satisfying results or can only estimate the temperatures of stator winding and rotor cage without stator core [2]. Other methods require powerful computation which can't be run on a resource limited node. All in all, none of them has been implemented on WSN so far.

We focus on the algorithm implementation on the wireless sensor network. The input signals are pre-processed in distributed wireless sensor nodes and transmitted to the host node, where the algorithm is implemented. In this article, section 2 gives a description of the system. The implementation of WTIM (wireless transducer interface module) and NCAP (network capable application processor) [3] are shown in section 3 and 4. The sequence of the network is described in section 5. Experiment results are discussed in section 6 and the conclusions follow in section 7.

2 The Proposed System Description

The state-space equations of the system are defined based on the thermal model [4]:

$$\frac{dT_{sw}}{dt} = \frac{-R_{sw}T_{sw}}{C_{sw}} + \frac{R_{sw}T_{sc}}{C_{sw}} + \frac{P_{sw}}{C_{sw}} \quad (1)$$

$$\frac{dT_{rc}}{dt} = \frac{-R_{rc}T_{rc}}{C_{rc}} + \frac{R_{rc}T_{sc}}{C_{rc}} + \frac{P_{rc}}{C_{rc}} \quad (2)$$

$$\frac{dT_{sc}}{dt} = \frac{-R_{sw}T_{sw}}{C_{sc}} + \frac{R_{rc}T_{rc}}{C_{sc}} + \frac{R_{sc}T_c}{C_{sc}} + \frac{(R_{sw}+R_{rc}+R_{sc})T_{sc}}{C_{sc}} + \frac{P_{sw}}{C_{sc}} \quad (3)$$

where subscript sw indicates the stator winding, rc for the rotor cage, sc for the stator core and c for the coolant air. T is the temperature above ambient, R is the thermal resistance, C is the thermal capacitance and P is the power loss of the machine. The losses P_{sw}, P_{rc}, P_{sc} can be calculated from the three-phase currents, voltages and rotor speed. The 4th-order KF algorithm has been implemented in the sensor node. The state vector is $\boldsymbol{x} = [T_{sw}, T_{rc}, T_{sc}, T_c]$ and the control vector is $\boldsymbol{u} = [P_{sw}, P_{rc}, P_{sc,}0]$. It has been proved that the temperatures can be estimated quite accurately both in simulation and in test bench. The detailed description of

the system and the KF algorithm can be found in the article [5].

2.1 Target System

The platform is the wireless sensor node Preon32 produced by Virtenio GmbH. It contains a 32-bit ARM Cortex-M3 micro-controller with 256kB flash memory for programming and 64kB RAM memory for data. A 2.4 GHz wireless transceiver, which is compliant to the IEEE 802.15.4 standard, can for example be used for ZigBee or 6LoWPAN communication. Two 12-bit analog-to-digital converters (ADC) with a maximum sampling rate of 1M samples/s are provided [6]. Its sampling period is derived from the CPU-clock and can be set with a resolution of 1 μs [7].

2.2 Structure and Topology of the System

Fig. 1 shows the structure of the temperature estimation system on WSN. Three Preon32 nodes are implemented as the WTIMs to acquire current, voltage, coolant air temperature and rotor speed. Another node is implemented as the NCAP to receive the data from different WTIMs and to process the KF algorithm for temperature estimation.

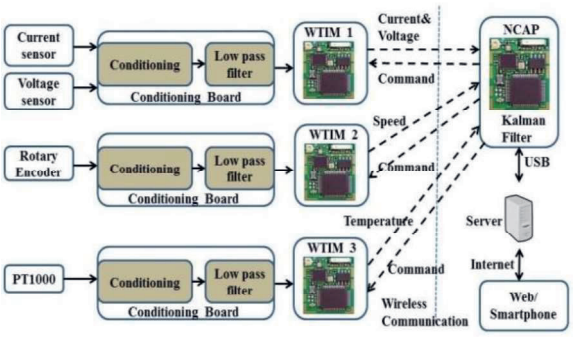

Fig. 1. Structure of the wireless sensor system.

3 Implementation of the Data Acquisition System in Distributed WTIMs

The data acquisition system (DAQ) is implemented based on the MSTL (MDT Smart Transducer Llibrary) which provides a universal interface to a variety of transducers and the implementation follows the IEEE1451 family of standards in many places. The *startTrigger* and *startStream* commands are broadcasted from NCAP to trigger the WTIMs simultaneously [8].

3.1 Analog Sensor Data Acquisition

Hall sensors are mounted on the conditioning board with low-pass filters to process analog three-phase currents and voltages [9]. A finite impulse response (FIR) filter with fixed-point arithmetic is implemented in the WTIMs for digital filtering. The sampling rate is 2000 Hz. The RMS of currents and voltages are calculated from a block of instantaneous value every 1 s. The mean value of the coolant air temperature T_c is calculated every 1 s from the sampled and filtered block, with a rate of 100 Hz and a block size of 10 samples/block. The correction coefficients of the sensors are stored in TEDS (Transducer Electronic Data Sheet) [3], making it possible to transfer the values to SI-units before transmission.

As data is acquired, filtered and transmitted continuously, the calculation time of the filter must be considered. In order to the data being processed and transmitted continuously, buffers are allocated using MEMB memory block allocators for the acquired data [10]. On the other hand, the computation time of the filter should be shorter than the acquisition time for one filtered block. The detailed signal processing time division is shown in Fig. 2:

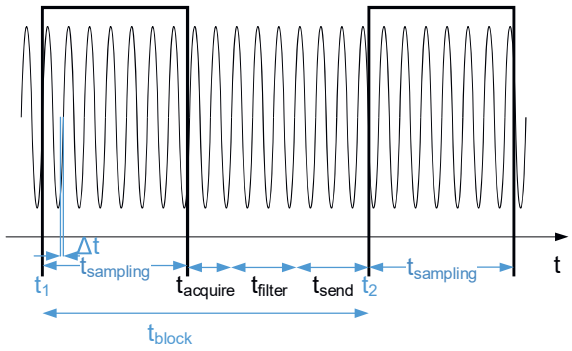

Fig. 2. Detailed processing time division.

The total acquisition and conversion time for one block (sampling time Δt is 50 μs with 8 channels and 16 repetition counts, totally 128 samples/block) is $t_{sampling} + t_{acquire} = 920\ \mu s$. The total filtering and sending time is $t_{filter} + t_{send} = 489\ \mu s$. As a result, the analog data acquisition system can process and transmit the data periodically.

3.2 Digital Sensor Data Acquisition

A rotary encoder (ROD 426B-6000) is mounted to the end of the machine shaft and connected to a conditioning board. A WTIM node is used to transfer the counts of pulse into the real rotor speed in rpm using *etimer* of Contiki.

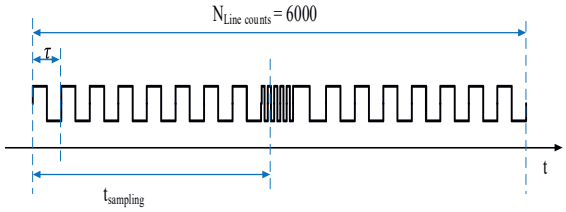

Fig. 3. The diagram of the generated pulses.

The formula to calculate the rotation speed in rpm from the pulses can be defined in equation 4, where τ is the time between two neighboring pulses, $N_{Line\ counts}$ is the number of encoder

lines per revolution, $t_{sampling}$ is the time period in one session, which is 12 degrees for the encoder.

$$Speed = \frac{60}{\tau N_{Line\ counts}} \quad (4)$$

3.3 Implemented Processes in WTIMs

The general structure of the implemented WTIM is shown in Fig. 4. The *IEEE1451.5* process is used to manage the radio module and to handle the communication of the WSN. The *IEEE1451.0* process is used to both manage the TEDS information and sample data of the sensors. Both rotation sensor and analog sensor acquisition systems are implemented. The values in SI units are sent back to the *IEEE1451.0* process periodically as soon as the WTIM receives *startTrigger* or *startStream* commands.

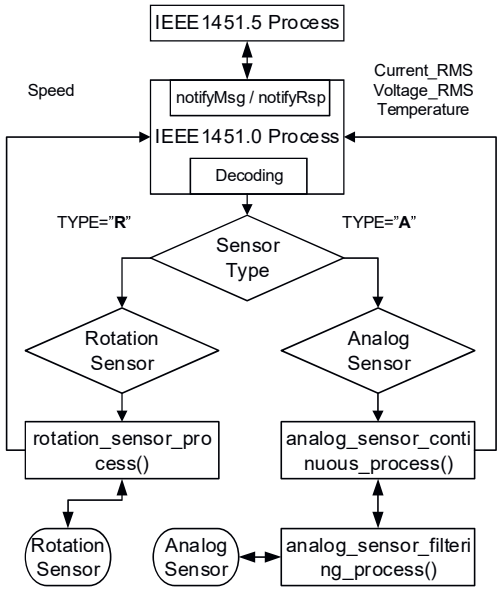

Fig. 4. The structure of implemented WTIMs.

4 Implementation of the Kalman Filter Algorithm in NCAP

This section provides the implementation of the KF algorithm based on the IEEE1451 standard in NCAP. The minimum implementation of the IEEE1451 standard has been integrated both in WTIM and NCAP. Sensors and actuators which are connected to WTIM can be managed by wireless commands from the NCAP. In our application, the KF algorithm is integrated in the NCAP to estimate the temperatures of stator windings, the rotor cage and the stator core of an induction machine. The Preon32 sensor node is resource restricted in respect to low-costs, weak power calculation and small memory size. In order to be implemented in the NCAP, the algorithm should be simple and efficient.

4.1 The Implementation of Processes in NCAP

Contiki OS is an event-driven system which is managed by protothreads. In order to operate different WTIMs, to manage the message transmission and to process the KF algorithm, several functional processes are implemented in the NCAP. The structures of the implemented processes are shown in Fig. 5:

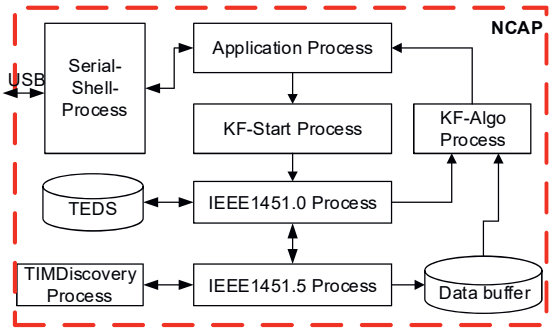

Fig. 5. The structure of implemented NCAP.

The *Serial-Shell* process was implemented for connecting the WSN to an arbitrary network. A PC connected to the NCAP works as a server of the network. Users can manage the WSN by using a web-based application or an App on a smart phone. *TIMDiscovery* process is used first for discovering WTIMs before every command from the application process. The *KF-Start* process is used for configuring and initiating. The *IEEE1451.5* process is implemented to manage the radio module and handle the data wireless transmission. The *IEEE1451.0* process represents the interlayer between the *IEEE1451.5* process and the *KF-Start* process. The buffer for storing data from different TIMs is allocated in this process. The received $I_{rms}, V_{rms}, \omega_r, T_c$ will be passed to the *KF-Algo* process for the temperature estimation and the results will be sent out through the *Serial-Shell* process.

4.2 KF Algorithm Implementation in the NCAP Using Fixed-Point Arithmetic

The KF algorithm for temperature estimation was first implemented in MATLAB. It was proved both in simulation and off-line experiments on a test bench that the temperatures can be accurately estimated. In order to implement the KF algorithm on the resource restricted sensor node, the same algorithm was implemented in the C programming language using floating point arithmetic on the Eclipse IDE platform. The workflow of the *KF-Algorithm* process is shown in Fig. 6 which can be summarized as follows: when *kf_process* starts, the system will retrieve and decode the messages from the *messages_buffer* where different messages

from different WTIMs are stored. The function *kf_data_gen()* is then called to calculate the losses P_{sw}, P_{rc}, P_{sc} from the rotor speed, the preprocessed RMS of currents and voltages, and generate the inputs with T_c, which are stored in the structure *kf_data{}*. Input data are passed to the *run_kf()* function where the main kalman filter process is performed. The DSP library is used for the fixed point matrix calculation. The state vector $\boldsymbol{X}$ and error covariance matrix $\boldsymbol{P}$ are stored in the structure *kf_filter{}* and sent back to the next recursion. The estimated temperatures T_{sw}, T_{rc}, T_{sc} are sent out for storage and display. T_{sw} is sent back to calculate the losses of the stator winding so that the resistance rising due to temperature can be compensated.

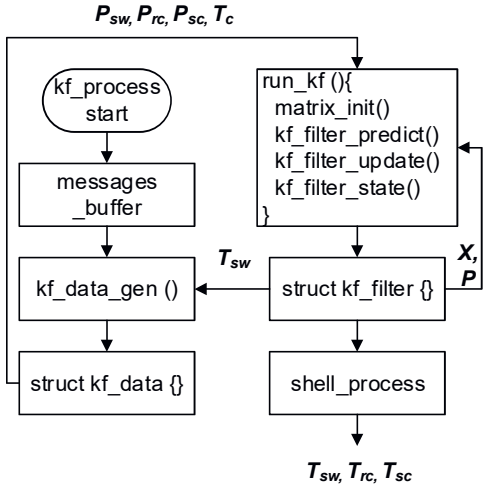

Fig. 6. The workflow of the KF algorithm process.

Compared to the implementation in MATLAB and Eclipse in C language, implementation on the Preon32 sensor node using Contiki OS creates several challenges.

Firstly, the methods to allocate and free memory space are different between the standard C library and Contiki OS. The standard C library allocates heap memory using the *malloc()* function. However, the Contiki platform specifies a small area of its memory space for the heap because of the resource restriction [9]. If the *malloc()* function were used for memory allocation, the heap could easily be overflowed. The MEMB memory block allocator is used to allocate a block of static memory to *struct kf_data{}*, which contains $P_{sw}, P_{rc}, P_{sc}, T_c$ as the input for the algorithm, and another to *struct kf_filter{},* which holds all the variables and matrixes which are used during the prediction stage and update stage of the KF algorithm.

The second challenge is that the preon32 does not have a floating point unit. It was clear that the floating point implementation cannot run on-line. As a result, fixed-point arithmetic is used for the implementation. In order to transfer the existing KF algorithm from floating point to fixed point representation, the proper Q-format (Qm.n) defined in [11] has to be considered first. Both the range and the resolution of the data are the key factors for choosing the type of Q-format. The system can avoid computation overflow by the saturation modes provided by CPUs, or by designing the arithmetic operations. The number of overflow checks was minimized by the division of the variables by 1000, which scaled all the variables and auxiliaries to [-1, $1-2^n$]. By checking the computation in MATLAB step by step, the minimum value of a number is 6×10^{-6}, which is larger than the Q0.31 format resolution. The data range and the resolution of variables are listed below in Tab. 1:

Tab. 1: The data range and the resolution of the variables.

Variable /1000	Max	Min	Resolution
Input	0.2303	-0.0006	6×10^{-6}
Output	0.4995	-0.1884	5×10^{-5}

Thus the Q0.31 format was used for the arithmetic with a resolution of 2^{-31} and a range of [-1, 0.999999999534]. This means that one bit is used to represent the sign within a two's complement, no bits represent the integer portion and the remaining 31 represent the fractional part of the number [11].

The third challenge is that the ARM Cortex-M3 processor provides the CMSIS DSP library, which contains matrix functions in fixed point [12]. The usage of specific matrix operation functions is listed in Tab. 2:

Tab. 2: The matrix operations function.

Functions	**Description**
arm_mat_init_q31	matrix initialization
arm_mat_add_q31	matrix addition
arm_mat_sub_q31	matrix subtraction
arm_mat_mult_q31	matrix multiplication
arm_mat_trans_q31	matrix transpose
mat_inv_q31	Matrix inverse

4.3 Memory Usage and Calculation Time

In the implementation of the KF algorithm in NCAP, all the memory blocks are allocated statically so that fragmentation can be avoided [4]. By using it this way, it is easy to analyze the memory (both RAM and Flash) usage. The

usage of RAM on the NCAP sensor node is shown in Fig. 7. The buffers of the KF algorithm take up about 24% of the total memory space. The basic system, which consists of the Contiki OS, the firmware provided by Virtenio, and other parts from the standard C library, consumes about 32%. The MSTL library takes up 7.4%. About 37% of the space is unused.

The usage of the flash memory for programming on the NCAP is shown in Fig. 8. Only about 5% of the memory is used for the KF algorithm and the MSTL library. The system takes up most of the used memory. The rest of about 62% of the total memory is not used.

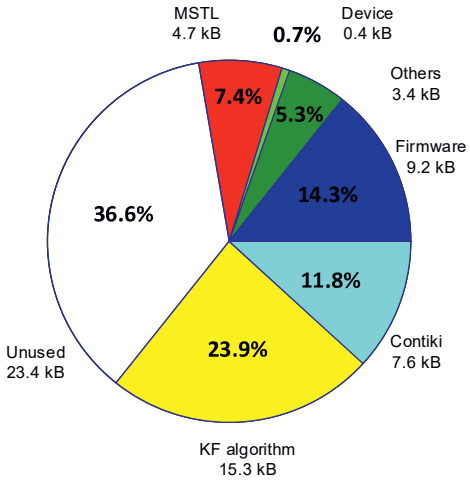

Fig. 7. The usage of the RAM on the NCAP sensor node (total memory: 64 kB).

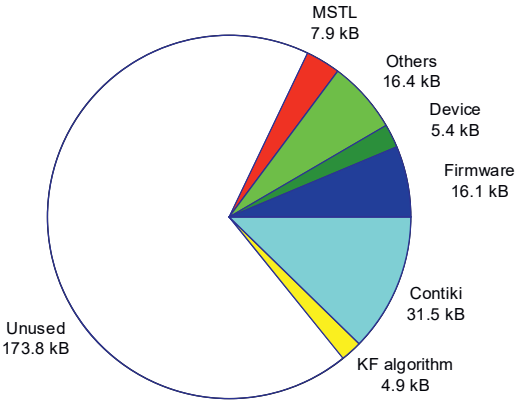

Fig. 8. The usage of flash memory on the NCAP sensor node (total memory: 256 kB).

The system gets the data from different buffers to generate the input, which costs 120 μs and the computation time of the KF algorithm for one step is about 600 μs. The total time of data generation and KF computation is much shorter than the calculation interval 1 s.

5 The Sequence of the WSN System

The sequence on the NCAP side is shown in Fig. 9. The *TIMDiscovery* command is first used to discover the available WTIMs in the network. After calling the *start_KF* function, the message is passed from the IEEE1451.0 layer to the IEEE1451.5 layer and then broadcasted to the WTIMs. Acquired data from different WTIMs is sent back to the NCAP and processed by the KF. Finally the temperatures are estimated.

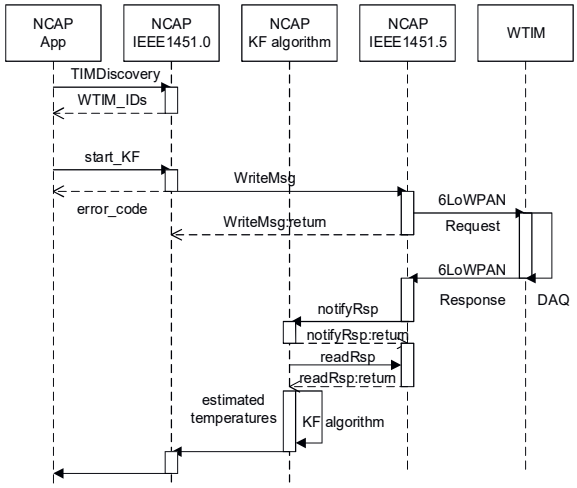

Fig. 9. The sequence on the NCAP side.

The sequence on the WTIM side is shown in Fig. 10. The *startStream* command is decoded for continuous data acquisition. The filtered and pre-processed data is sent back to the NCAP for the KF algorithm.

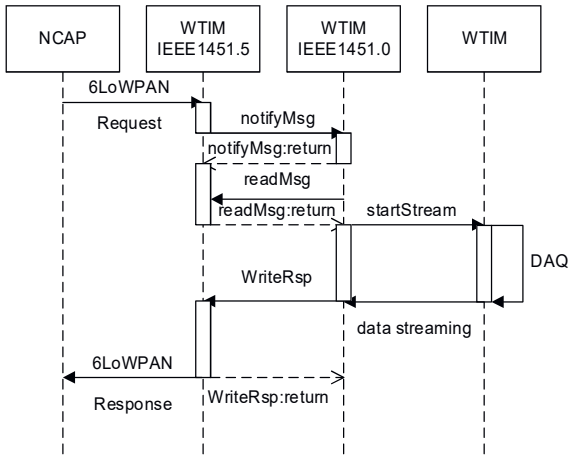

Fig. 10. The sequence on the WTIM side.

6 Experiments

Two experiments were performed on the test bench (Siemens machine: 1 LA5107-4AA20) using the WSN temperature estimation system. Fig. 11 shows the continuous full-load test S1 and Fig. 12 shows the intermittent-load S6 (6 minutes no load followed by 4 minutes full load). All the temperatures were estimated

accurately under S1 and S6 with a maximum error of 2 K, with the accuracy of 97%, except for the rotor cage temperature error of 3.8 K under S6. It's due to the installation of PT1000 on the rotor cage, which influences the flux density and generates excessive losses (about 55 w) compared to a healthy machine [13].

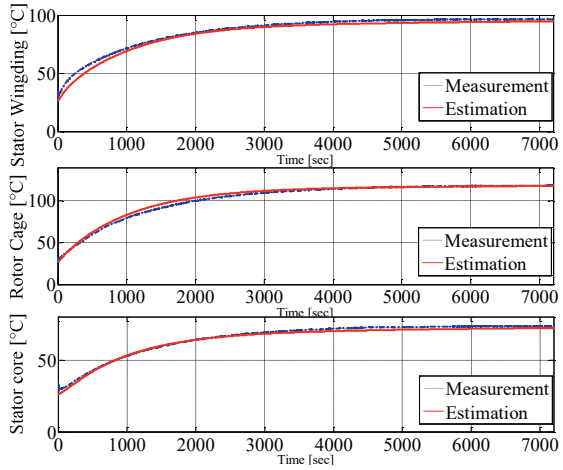

Fig. 11. Comparison of measured and estimated temperatures under S1.

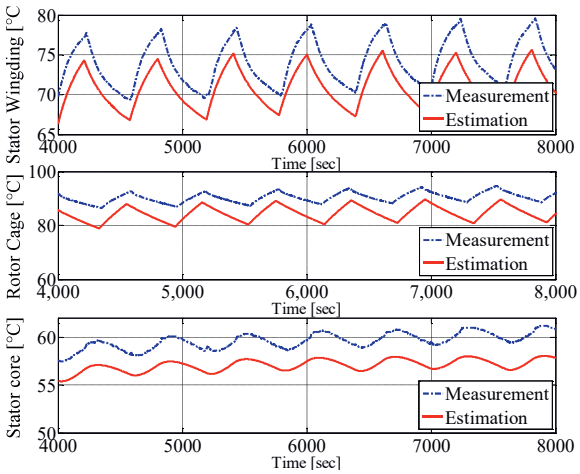

Fig. 12. Comparison of measured and estimated temperatures under S6.

7 Conclusions

This paper describes the implementation of the temperature estimation system of induction machines on a WSN. The 4th-order KF with fixed-point arithmetic was implemented in NCAP. Three WTIMs were implemented as the data acquisition systems. Compared to the floating point implementation, the fixed point had the same estimation accuracy at only about one-fifth of computation time. The KF algorithm is independent from the control strategy and the running conditions. That means no matter what the rotor speed is, and what the mechanical load is, as long as there are currents through the stator winding, the temperature can be estimated correctly. The experiments prove that the KF implementation is suitable for real-time temperature estimation on a resource limited wireless sensor node.

References

[1] M. O. Sonnaillon, G. Bisheimer, C. D. Angelo, and G. O. Garca, "Online sensorless induction motor temperature monitoring," IEEE Transactions on Energy Conversion, vol. 25, no. 2, pp. 273–280, June 2010.

[2] G. M. B. S. Ozsoy, E.E., "Simultaneous rotor and stator resistance estimation of squirrel cage induction machine with a single extended kalman filter," Turk. J. Elec. Eng. & Comp. Sic., 2010.

[3] IEEE standard for a smart transducer interface for sensors and actuators – common functions, communication protocols, and transducer electronic data sheet (teds) formats. IEEE Std 1451.0-2007, pages 1–335, Sept 2007.

[4] A. Haumer, C. Kral, V. Vukovic, A. David, C. Hettfleisch, and A. Huzsvar, "A parametrization scheme for high performance thermal models of electric machines using modelica," in MATHMOD VIENNA, 2012.

[5] Y. Huang, C. Gühmann, "Wireless Sensor Network for Temperature Estimations in an Asynchronous Machine Using a Kalman Filter", 15th IMEKO TC10 Workshop on Technical Diagnostics, 2017 (accepted not published yet).

[6] Virtenio GmbH: http://www.virtenio.com/en/

[7] J. Funck and C. Guehmann. "A flexible filter for synchronous angular resampling with a wireless sensor network". Measurement, 98:393 – 406, 2017.

[8] IEEE standard for a smart transducer interface for sensors and actuators – common functions, communication protocols, and transducer electronic data sheet (teds) formats. IEEE Std 1451.5-2007, pages C1–236, Oct 2007.

[9] T. Hopp, J. Funck. "Intelligenter Sensor zur Leistungsmessung im Dreiphasennetz". Master thesis, TU Berlin, May 2013.

[10] Contiki: https://github.com/contiki-os/contiki/wiki/Memory-allocation

[11] A. Bock, D. Liu, J. Funck, A. Giedymin, R. Burke C. Gühmann, "Wireless Sensor for Temperature and Flux Measurements in an Axial Flux Machine", AMA Conference 2015-SENSOR and IRS^2 2015.

[12] CMSIS:http://www.arm.com/products/processors/cortex-m/cortexmicrocontroller-software-interface-standard.php.

[13] J. F. Bangura, N. A. Demerdash, "Effects of broken bars/end-ring connectors and airgap eccentricities on ohmic and core losses of induction motors in asds using a coupled finite element-state space method", IEEE Trans. Energy Convers., vol. 15, no. 1, pp. 40-47, Mar. 2000.