computing@computingonline.net
www.computingonline.net

Print ISSN 1727-6209
On-line ISSN 2312-5381
International Journal of Computing

A METHOD OF PREDICTING THE MAINTENANCE PERIOD OF EMBEDDED SYSTEMS FOR PREVENTING BREACH OF THEIR TIME REQUIREMENTS

Dmytro Fedasyuk, Tetyana Marusenkova, Ratybor Chopey

Lviv Polytechnic National University, 12 Bandery str., 79013, Lviv, Ukraine, http://www.lp.edu.ua
fedasyuk@gmail.com, tetyana.marus@gmail.com, chopey.ratybor@gmail.com

Paper history:
Received 14 February 2018
Received in revised form 24 April 2018
Accepted 04 May 2018
Available online 30 June 2018

Keywords:
real-time embedded system;
predicting maintenance period;
firmware execution time;
hardware aging.

Abstract: The work deals with a significant problem of ensuring that the execution time of a firmware running inside a microcontroller-based real-time embedded system never goes out of its expected range, no matter for how long the embedded system has been used. Once having been tested before the first usage, a newly created embedded system is gradually getting slower in its response, due to the fact that its hardware components get worn-out with aging. A possible solution is a replacement of the hardware components that most contribute to such a change in the response time of the embedded system. If such a replacement takes place too far in advance, long before hardware components actually start showing any decline in their response time, the above-mentioned solution is cost-ineffective and impractical, as it leads to a waste of equipment and efforts. We introduce a method for predicting the appropriate maintenance period of a real-time embedded system on the basis of the characteristics of its hardware components.

Copyright © Research Institute for Intelligent Computer Systems, 2018.
All rights reserved.

1. INTRODUCTION

The firmware execution time is one of the most important metrics of software running in real-time embedded systems, as the applicability of the latter depends not only on the logical correctness of such a software, but also on the timeliness of its results [1]. If a real-time embedded system fails to accomplish its time-critical functions on time, it might lead to catastrophic consequences, thus a breach of time requirements can be considered as the failure of a system.

Thus, in order to ensure the required level of reliability and safety of real-time embedded systems, additional kinds of software analysis should be applied to them. One of such kinds is an analysis of the firmware execution time controlled by standards DO178B [2] and ARINC 653 [3]. However, these standards do not regulate any conditions for testing and do not prescribe any methods that would make allowances for the fact that hardware components of an embedded system are gradually getting worn-out over time and, for this reason, the response time of any embedded system tends to worsen over time (the longer an embedded system has been in use, the slower its response time proves to be).

One of the ways to keep the firmware execution time within its expected range is to perform the maintenance of an embedded system on time, i.e., to replace all the hardware items composing the system, that tend to show some noticeable decline in their response time with age.

If a replacement of hardware components has been done too early, long before any considerable change in the firmware response time takes place, the procedure proves to be a waste of components, costs for the delivery of an embedded system under maintenance to the manufacturer's office, and human resources responsible for such a replacement. On the other hand, if a replacement takes place too late, the whole embedded system does not respond within an expected time period any longer, i.e., fails to function properly.

In [4] the approach and modeling technique for estimation availability function of FPGA based-systems using a two different strategy of maintenance was presented. However, this approach can be used for estimating embedded system that is based on microcontroller.

The work is aimed at the development of a method that would enable a manufacturer of a real-

time embedded system to determine the most reasonable maintenance period, i.e., such a maintenance period that is free of a waste of time and costs attendant to replacing old specimens of hardware components with new ones and, at the same time, enables the manufacturer to ensure that the execution time of the firmware running in the embedded system never exceeds its maximum allowed value.

2. RELEVANT WORKS

All the modern methods for analysis of the firmware execution time fall into two large groups in accordance with the main principle of their execution: static methods and dynamic methods.

Static methods do not assume any real firmware execution, i.e. the firmware runs neither in an embedded system nor in an emulator during static analysis. All methods of this group use the source code of the firmware in question and/or the model of the hardware architecture of the embedded system being considered.

Since the architectures of processors are constantly evolving, there is a constant need in modification of the existing models so that they keep applicable. Thus, when a new processor hardware model is developed, those who work on modification of static models focus their attention on the structure of the cash memory and the analysis of the content of the cash [5-7], the structure of the instruction pipeline [8, 9], and losses in data rate during transmitting data via communication interfaces [10]. However, the more sophisticated the architectures of modern processors become, the more efforts and time their analysis and modification of the models require.

Moreover, static methods never take into consideration the influence of the environment on the firmware execution time. Besides, static methods do not make allowances for the dependence of the firmware execution time on the age of the underlying hardware components.

Dynamic methods for analysis of the firmware execution time are based on the actual measurement of the response time of the firmware running inside a real embedded system or a simulator. Due to their close connection with the real execution of firmware, they are considered to be potentially more accurate than static methods, for any measured execution time does include the influence of different ambient factors implicitly. Thus, a dynamic method makes allowances for a variety of factors influencing the firmware execution time and not only the source code itself. Moreover, in most cases measurement of the execution time is performed in real working conditions, but not in some artificial, laboratory environment, completely different from the real one. Typically, dynamic methods are applied to measure the firmware execution time using a logical analyzer or an oscilloscope [11, 12], hardware tracing [13, 14], integration of an additional code into an embedded system being tested [15, 16] or simulators of an embedded system [17]. Each of these approaches has their advantages and disadvantages. However, neither of dynamic methods takes into consideration the fact that the execution time of any firmware running inside a real embedded system might change because the underlying hardware components change their behavior with age. Thus, these dynamic methods provide one-shot results for an embedded system either before it started being used or at the very beginning of its life cycle, but no predictions were made what would happen to the firmware execution time with age.

All the above said statements lead to the idea of developing a method that would promote to determine the maintenance period to ensure that the execution time of the firmware running in the embedded system never exceeds its maximum allowed value.

3. CONCEPT CONSTRUCT AND TASK FORMULATION

In order to predict the execution time of a program code, we represent the code as a control flow graph (Fig. 1).

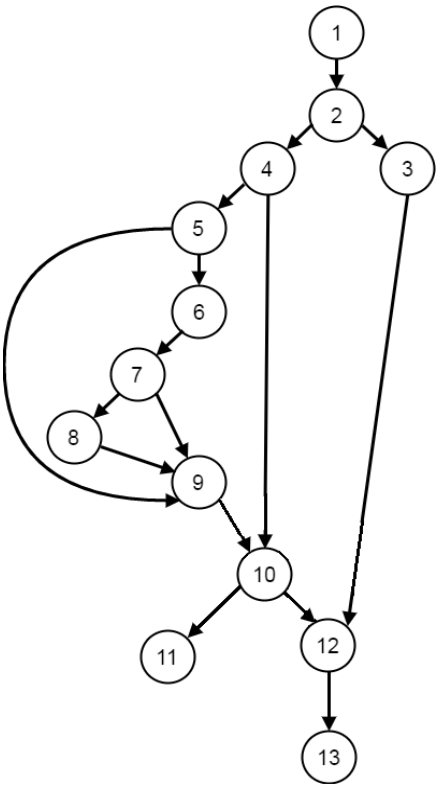

Figure 1 – An example of a control flow graph that represents a function in a program.

The vertices of a control flow graph denote operators or function calls. The edges represent connections between vertices.

The execution time of any branch in a program (τ_b) can be calculated by the following formula:

$$\tau_b = \sum_{i=1}^{n} \tau_i , \qquad (1)$$

where τ_i is the time spent on vertex i (i.e., the time spent on execution of instruction i), n is the total number of vertices in the branch.

The total execution time of a program branch depends not only on the duration of an instruction but on the response time of different peripheral devices as well, and the latter tend to get slower in their response over time.

Taking into account this fact, we can expand formula (1) as follows:

$$\tau_b = \sum_{j=1}^{k} \tau_j + \sum_{i=1}^{l} \tau_i(T) , \qquad (2)$$

where τ_j is the time spent on vertex j (the time of execution of the corresponding instruction), k is the number of vertices that take always the same amount of time to be executed (the corresponding instructions do not depend on any peripheral devices, but on the microcontroller's computational resources only), τ_i is the time spent on execution of an instruction that depends on the response time of a peripheral device (the response time changes with the age of a peripheral device), T is the time period of the embedded system's being in use, l is the number of vertices such that the time spent on them depends on the response time of peripheral devices composing the embedded system.

Thus, we came to the conclusion that there exists a lack of such a method for estimation of the firmware execution time that would make allowances for the fact that the response time of peripheral devices changes over the time of their use. Such a method would allow calculation of the maintenance period of an embedded system in question.

4. MATHEMATICAL MODELS

4.1 INPUT DATA

In accordance with the technical report [18], we established the dependence of the response time of a hardware component on the time during which it has been in use. The results are summarized in Table 1.

Table 1. The influence of aging of a peripheral device on the response time.

Time of being in use, years	The difference between the actual value of the response time of a peripheral device and its nominal value, %
0	0
0.25	10
0.5	10
0.6	10
1.7	10
1.9	15
2.2	15
3	18

The nominal value of the response time is equal to $7.812 \cdot 10^{-3}$ s [19]. The results of calculation of the response time of a peripheral device on the basis of the data from Table 1, are presented in Table 2.

Table 2. The dependence of the response time on how long the system has been in use.

The time of being in use, years	The response time of a peripheral device, s
0	$7.8125 \cdot 10^{-3}$
0.25	$8.59375 \cdot 10^{-3}$
0.5	$8.59375 \cdot 10^{-3}$
0.6	$8.59375 \cdot 10^{-3}$
1.7	$8.59375 \cdot 10^{-3}$
1.9	$8.984375 \cdot 10^{-3}$
2.2	$8.984375 \cdot 10^{-3}$
3	$9.21875 \cdot 10^{-3}$

The dependence of the response time on the time of being in use, drawn on the basis of the experimental data, is shown in Fig. 2.

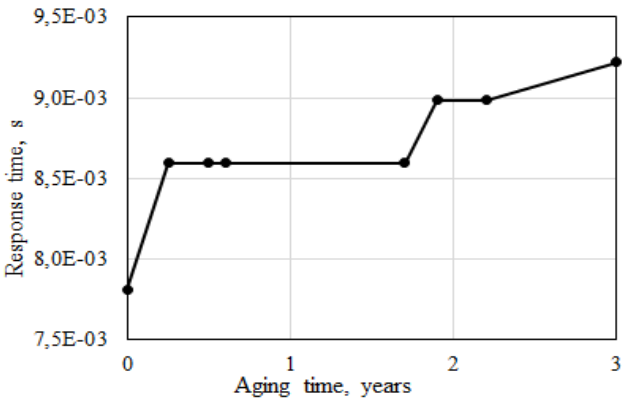

Figure 2 –The dependence of the response time on the time of being in use.

Having analyzed the above-stated data, one can conclude that the analytical dependence could be approximated well enough using regression methods

including methods of linear, cubic and logarithmic regression.

4.2 CHOSING AN APPROPRIATE MODEL

In order to calculate the response time of a peripheral device using the linear regression method, we apply the following formula:

$$\hat{\tau} = aT + b, \tag{3}$$

where T is the time period during which the peripheral device has been used, and a and b are some coefficients that can be evaluated using formulas (4) and (5):

$$a = \frac{\sum_{i=1}^{n} T_i \sum_{i=1}^{n} \tau_i - n \sum_{i=1}^{n} T_i \tau_i}{\left(\sum_{i=1}^{n} T_i \right)^2 - n \sum_{i=1}^{n} T_i^2}, \tag{4}$$

$$b = \frac{\sum_{i=1}^{n} T_i \sum_{i=1}^{n} T_i \tau_i - \sum_{i=1}^{n} T_i^2 \sum_{i=1}^{n} \tau_i}{\left(\sum_{i=1}^{n} T_i \right)^2 - n \sum_{i=1}^{n} T_i^2}. \tag{5}$$

Using formulas (4) and (5), we can figure out that the values of the coefficients a and b for approximation by the linear regression method are as follows:

$$a = 0.0003, \quad b = 0.0083. \tag{6}$$

In order to evaluate the response time of a peripheral device using the cubic regression method, we apply the following formula:

$$\hat{\tau} = aT^3 + bT^2 + cT + d, \tag{7}$$

where a, b, c and d are some coefficients that can be evaluated when solving the following equation set:

$$\begin{cases} a\sum_{i=1}^{n} T_i^3 + b\sum_{i=1}^{n} T_i^2 + c\sum_{i=1}^{n} T_i + nd = \sum_{i=1}^{n} \tau_i, \\ a\sum_{i=1}^{n} T_i^4 + b\sum_{i=1}^{n} T_i^3 + c\sum_{i=1}^{n} T_i^2 + d\sum_{i=1}^{n} T_i = \sum_{i=1}^{n} T_i \tau_i, \\ a\sum_{i=1}^{n} T_i^5 + b\sum_{i=1}^{n} T_i^4 + c\sum_{i=1}^{n} T_i^3 + d\sum_{i=1}^{n} T_i^2 = \sum_{i=1}^{n} T_i^2 \tau_i, \\ a\sum_{i=1}^{n} T_i^6 + b\sum_{i=1}^{n} T_i^5 + c\sum_{i=1}^{n} T_i^4 + d\sum_{i=1}^{n} T_i^3 = \sum_{i=1}^{n} T_i^3 \tau_i; \end{cases} \tag{8}$$

Using formula (8) we evaluate the values of the coefficients a, b, c and d for approximation by the cubic regression method:

$$a = 0.0002, \; b = -0.001, \; c = 0.0017, \; d = 0.008. \tag{9}$$

The response time of a peripheral device using the logarithmic regression method can be calculated by the applying the following formula:

$$\hat{\tau} = a + b \ln T, \tag{10}$$

where a and b are some coefficients that have to be calculated using formulas (11) and (12).

$$b = \frac{n \sum_{i=1}^{n} (\tau_i \ln T_i) - \sum_{i=1}^{n} \ln T_i \cdot \sum_{i=1}^{n} \tau_i}{n \sum_{i=1}^{n} \ln^2 T_i - \left(\sum_{i=1}^{n} T_i \right)}, \tag{11}$$

$$a = \frac{1}{n} \sum_{i=1}^{n} \tau_i - \frac{b}{n} \sum_{i=1}^{n} \ln T_i. \tag{12}$$

Using formulas (11) and (12), we have figured out that the values of coefficients a and b for approximation by the logarithmic regression methods are as follows:

$$a = 0.0083, \quad b = 1.0391. \tag{13}$$

4.3 THE INACCURACY OF THE MODELS

In order to choose the most appropriate regression model, we have to verify how the results provided by each of the above-mentioned models fit the experimental data. In order to do this, we first visualize the analytical dependencies obtained on the basis of the coefficients, calculated previously, and visually compare the resulting curves with the graph built on the basis of the experimental data.

Then, we choose a number of equal-distant points on the time axis that did not participate in either of the above-mentioned equation sets used for calculation of coefficients a and b. For each of the chosen time points, we evaluate the difference between the experimental data and the results provided by each of the regression methods.

The average approximation error is calculated by the formula:

$$\overline{A} = \frac{1}{n} \sum \left| \frac{\tau_i - \hat{\tau}_i}{\tau_i} \right| \cdot 100\%. \tag{14}$$

Using formula (14) we calculated the average approximation errors:

for linear regression method:

$$\overline{A} = 2.05\%, \quad (15)$$

for cubic regression method:

$$\overline{A} = 1.53\%, \quad (16)$$

for logarithmic regression method:

$$\overline{A} = 1.64\%. \quad (17)$$

For visual comparison of the obtained results, we show a graph depicting the experimental data and the approximation results obtained by the linear, cubic and logarithmic regression methods, Fig. 3.

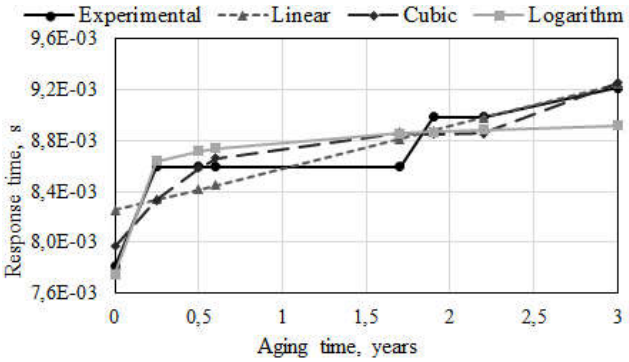

Figure 3 – The experimental data and analytical dependencies of the response time of a peripheral device on the time of being in use.

As one can see on the graph, the cubic regression method provides the least approximation inaccuracy (on average). Thus, we chose this method for predicting the firmware execution time and the right time moments of embedded systems' maintenance.

5. EXPERIMENTS

In order to verify our ideas about the importance of taking into account the aging effects of electronic units for predicting an appropriate maintenance period of an embedded system, we chose a commercial product, the main functions of which include regular measurement of the temperature and pressure inside industrial pumps, logging all the values of the temperature and pressure lower or higher than their allowed lower an upper limits correspondingly, and control of valves on the basis of the temperature and pressure readings.

The firmware thread responsible for measurement is shown in Fig. 4.

```
void MeasureTask(void const *argument)
{
  osEvent evt;
  SYNC_BUFF_T SaveData;
  float SampleBuff[3];
  float intrnal,temp,PressureVal;

  for (;;)
  {
    evt = osSignalWait(0x0001, osWaitForever);
    if (evt.status == osEventSignal)
    {
      /* Clear data buffers */
      memset(SampleBuffer, 0x00, sizeof(SampleBuff));
      memset(SaveData, 0x00, sizeof(SYNC_BUFF_T));

      /* Read temperature from Ch1 of external ADC */
      internal = ADS_Read(ADSCON_CH1);
      temp = local_compensation(internal);
      SampleBuff[0] = ADC_code2temp(temp);

      /* Read temperature from Ch2 of external ADC */
      internal = ADS_Read(ADSCON_CH2);
      temp = local_compensation(internal);
      SampleBuff[1] = ADC_code2temp(temp);

      /* Read internal temperature of external ADC */
      internal = ADS_Read(ADSCON_INTERNAL);
      temp = local_compensation(internal);
      SampleBuff[2] = ADC_code2temp(temp);

      /* Send command for stert conversion on internal ADC */
      StartConversionOfInternalADC1();
      /* Calculate the pressure value */
      PressureVal = ReadPressValFromInternalADC1();

      /* Copy timestamp inside data structure */
      memcpy(&SaveData.Time, &CurrTime, sizeof(TIME_T));
      /* Copy measured values inside data structure */
      memcpy(&SaveData.Temp, &SampleBuff, sizeof(SampleBuff));
      SaveData.Perss = PressureVal;
      /* Calculating the checksum */
      SaveData.crc = CalculateCRC(SaveData);
      /* Save data on the external memory */
      WriteToExternalMemory(&SaveData, position);
      /* control of the valves according to read
      values of temperature and pressure */
      Valve_control(SampleBuff);
      osThreadYield();
    }
  }
}
```

Figure 4 – An example of a time-critical thread in firmware.

In accordance with the user requirements, the thread responsible for measurement should be executed each 100 ms, and the maximum execution time of the thread's fragment intended for measurement and saving data, should not exceed 70 ms.

The thread used here as an example contains functions of two types:

- Functions, the execution time of which depends only on the computational capabilities of the underlying microcontroller (osSignalWait, memset, local_compensation, ADC_code2temp).
- Functions, the execution time of which depends on the response time of peripheral devices (the execution time of function ADS_Read() depends on the response time of an external ADC ASD1118. The execution time of function WriteToExternalMemory()

depends on the response time of an external flash memory. The execution time of Valve_control function depends on the execution time of the module responsible for controlling the valves.

In order to predict the execution time of a firmware fragment, we find out that the nominal value of the response time for external ADC ADS1118 is equal to 15.625 mcs, in accordance with the device's datasheet [19].

From the datasheet [20] of flash memory AT45DB041E we fetch the nominal value of its response time (0.5 mcs).

Here we assume that the dependence of the response time on the time of being in use is similar to that one presented in Table 1.

We have calculated the response time of the peripheral devices in the reference points. The calculation results are given in Tables 3 and 4.

Table 3. The dependence of the response time of ADC ADS1118 on the time during which it has been in use.

Time of being in use, years	The response time of a peripheral device, s
0	$15.625 \cdot 10^{-3}$
0.25	$17.188 \cdot 10^{-3}$
0.5	$17.188 \cdot 10^{-3}$
0.6	$17.188 \cdot 10^{-3}$
1.7	$17.188 \cdot 10^{-3}$
1.9	$17.969 \cdot 10^{-3}$
2.2	$17.969 \cdot 10^{-3}$
3	$18.438 \cdot 10^{-3}$

Table 4. The dependence of the response time of Flash AT45DB041E on the time during which it has been in use.

Time of being in use, years	The response time of a peripheral device, s
0	$0.5 \cdot 10^{-3}$
0.25	$0.55 \cdot 10^{-3}$
0.5	$0.55 \cdot 10^{-3}$
0.6	$0.55 \cdot 10^{-3}$
1.7	$0.55 \cdot 10^{-3}$
1.9	$0.575 \cdot 10^{-3}$
2.2	$0.575 \cdot 10^{-3}$
3	$0.59 \cdot 10^{-3}$

Using formula (8), we calculate the following:

1) The approximation coefficients for calculating the response time of ADC ADS1118.

$$a = 0.0004, b = -0.0021, c = 0.0034, d = 0.0159. \quad (18)$$

2) The approximation coefficients for calculating the response time of flash memory AT45DB041E.

$$a = 0,\ b = -0.0001,\ c = 0.0001,\ d = 0.0005. \quad (19)$$

Using the approach described in [10], we estimate the execution time of each function that depends only on the computational resources of the underlying microcontroller.

In accordance with formula (2), we evaluate the execution time of the fragment of the thread in firmware intended for measuring and logging data. The calculation results of the execution time with consideration of the aging effect and without such consideration are shown in Fig. 5.

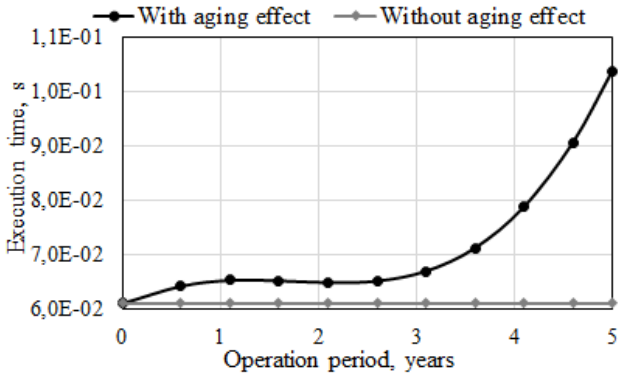

Fig. 5 – The execution time of a software code that taking into account the aging effect and without it.

The obtained results demonstrate that if one ignores the influence of the aging effect of a peripheral device on the response time of the latter particularly and the firmware execution time in general, one is likely to underestimate the execution time of the whole embedded system's firmware drastically, and such an underestimation might have catastrophic consequences in real-time embedded systems.

Making allowances for the influence of the aging effect on the firmware execution time would enable us to predict the right time for the maintenance of an embedded system before the latter becomes inapplicable because it fails to perform its time-critical tasks quickly enough.

For example, on the basis of the obtained results, we came to the conclusion that in order to ensure that a specific firmware fragment meets its required execution time, one should replace the worn-out components in the embedded system in question after three years of being in use.

6. DISCUSSION OF THE RESULTS

The research carried out by the authors show that the least approximation inaccuracy can be achieved due to the cubic regression method, thus one needs to choose the cubic approximation when predicting the suitable maintenance period.

It's worth bearing in mind that in order to calculate the approximation coefficients with the minimum inaccuracy we need to use the response time of a peripheral device, not the difference between the nominal value of the response time and the actual value.

For this reason, one should evaluate the approximation coefficients for each peripheral device individually, even if different peripheral devices are characterized by the same change of their response time with respect to the nominal value in percent, with age. For example, device 1 responses within 1 ms, device 2 – within 5 ms and device 3 – within 7 ms and the response time of each of these devices changes by 10% after three months and by 15% after six months of being in use. Despite the fact that it would be convenient to operate with relative values, which are identical for the three devices, we still should take the absolute value of the response time of each device individually.

Significant advantage is in the fact that input data for proposed method can be obtained using any static or dynamic method for analysis of the firmware execution time, which was represented in chapter 2.

7. CONCLUSION AND FUTURE WORK

In this work, we have solved an important problem of determining the most appropriate time moments for maintaining an embedded system in order to ensure that the execution time of the firmware running in this system is kept within its required range and the system is still able to perform its time-critical tasks on time.

The approximation coefficients presented in this paper are unique for the hardware component taken as an example to show the significance of the aging effect in electronic units in general. In order to apply the proposed method for predicting the most appropriate maintenance period for an arbitrary microcontroller-based real-time embedded system, one should take the following steps. First, one is supposed to detect the hardware components that tend to change their behavior over time and apply the cubic regression method in order to draw the analytical dependency of the hardware component's response time on the time period, during which it has been in use. One should make sure that the analytical dependency is correct by evaluation the average approximation error. Then, one finds the maximum time value at which the execution time calculated using formula (2) still does not exceed the allowed limit.

Our further planned research is aimed at automating the proposed approach and integrating the module that implements it into a software tool intended for automated testing of the execution time of firmware running in embedded systems.

7. REFERENCES

[1] J. Stankovic, "Misconceptions about real-time computing: a serious problem for next generation systems," *Computer*, Vol. 21, Issue 10, pp. 10-19, 1988. DOI: 10.1109/2.7053

[2] T. K. Ferrel, U. D. Ferrel, *RTCA DO-178B/EUROCAE ED-12B,* Ferrell and Associates Consulting.

[3] *ARINC 653 - An Avionics Standard for Safe, Partitioned Systems*. Wind River Systems, IEEE Seminar, 2008.

[4] V. Kharchenko, Y. Ponochovnyi, A.-S. M. Q. Abdulmunem and A. Boyarchuk, "Security and availability models for smart building automation systems", *International Journal of Computing*, Vol. 16, Issue 4, pp. 194-202, 2017.

[5] X. Li, A. Roychoudhury, T. Mitra, "Modeling out-of-order processors for software timing analysis," *in Proceedings of the 25th IEEE Real-Time Systems Symposium (RTSS'04)*, Libon, December 5-8, 2004, pp. 92-103. DOI: 10.1109/REAL.2004.33.

[6] J. Souyris, E. Pavec, G. Himbert, "Computing the worst-case execution time of an avionics program by abstract interpretation," *in Proceedings of the 5th International Workshop on Worst-Case Execution Time Analysis, (WCET'2005),* Palma de Mallorca, July 5, 2005, pp. 55-59. DOI:10.4230/OASIcs.WCET. 2005.810.

[7] C. Ferdinand, R. Heckmann, H. Theiling, "Convenient user annotations for a WCET tool," *in Proceedings of the 3-rd International Workshop on Worst-Case Execution Time Analysis, (WCET'2003)*, Porto, July 1, 2003. pp. 17–20.

[8] J. Engblom *Processor Pipelines and Static Worst-Case Execution Time Analysis*: Ph.D. thesis / J. Engblom – Uppsala: Uppsala University, 2002. ISBN 91-554-5228-0.

[9] C. Healy, R. Arnold, F. Muller, "Bounding pipeline and instruction cache performance," *IEEE Transactions on Computers*, Vol. 48, Issue 1, pp. 53-70, 1999. DOI: 10.1109/12.743411.

[10] P. Atanassov, R. Kirner, P. Puschner, "Using real hardware to create an accurate timing model for execution-time analysis," *in Proceedings of the IEEE Real-Time Embedded Systems Workshop, held in conjunction with (RTSS'2001)*, 2001.

[11] D. Stewart, "Measuring execution time and real-time performance," *in Proceedings of the Embedded Systems Conference (ESCSF)*, San Francisco, 2004.

[12] Y. Zhang, *Evaluation of methods for dynamic time analysis for CC systems AB*, Thesis of Master's degree, Vasteras, Malardalen University, 2005.

[13] M. Wahler, E. Ferranti, R. Steiger, R. Jain, "CAST: automating software tests for embedded systems," *in Proceedings of the 15th International Conference on Software Testing, Verification and Validation,* April 17-21, 2012, pp. 123-133. DOI: 10.1109/ICST.2012.126.

[14] R. Chopey, B. Knysh, D. Fedasyuk, "The model of software execution time remote testing," *in Proceedings of the 9th International Conference of Young Scientists «Computer Science and Engineering 2017» (CSE'2017)*, Lviv, 2017, pp. 398–402.

[15] R. Kirner, "The WCET Analysis Tool CalcWcet167," *in Proceedings of the 5th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies,* October 15-18, 2012, pp. 158-172. DOI: 10.1007/978-3-642-34032-1_17.

[16] D. Fedasyuk, R. Chopey, B. Knysh, "Architecture of a tool for automated testing the worst-case execution time of real-time embedded systems' firmware," *in Proceedings of the 14th International Conference of Experience of Designing and Application of CAD Systems in Microelectronics (CADSM),* Lviv, February 21-25, 2017, pp. 278–282. DOI: 10.1109/CADSM.2017.7916134

[17] J. Engblom, F. Stappert, A. Ermedahl, "Structured testing of worst-case execution time analysis tools," *in Proceedings of the 21st Real-Time System Symposium (RTSS/WIP'00)*, Orlando, November 27-30, 2000, pp. 154–163.

[18] U.S. Nuclear Regulatory Commission, *Effect of Aging on Response Time of Nuclear Plant Pressure Sensors*, Washington DC, 1989.

[19] *ADS1118 Ultrasmall, Low-Power, SPI™-Compatible, 16-Bit Analog-to-Digital Converter with Internal Reference and Temperature Sensor*. Texas Instruments, 2013.

[20] *AT45DB041E 4-Mbit DataFlash SPI Serial Flash Memory*. Adesto Technologies, 2013.

Dmytro V. Fedasyuk *was born in 1955. In 2000 he became a doctor of science after defending his work in Lviv. In 2002 he was promoted to the academic rank of professor. His scientific contribution is contained in four monographs and over 250 works published in a wide range of scientific journals included those well-known all over the world.*

His main fields of interest are mathematical modeling; modeling and analysis of thermo-electrical processes in microelectronic systems, Internet technologies, software design.

Tetyana A. Marusenkova *was born in 1982. In 2011 she joined Software Department of Lviv Polytechnic National University as a teacher. In 2014 she started working in the team built of teachers and students of Software department in order to develop embedded systems in collaboration with Dinamica Generale S.p.A., an Italian company providing modern electronic solutions and sensors.*

She is a co-author of over 50 papers and proceedings.

Ratybor S. Chopey *is a postgraduate student of the Software Department of Lviv Polytechnic National University. He received his master degree in radio-frequency engineering in Lviv Polytechnic National University in 2014.*

He is a co-author of 7 papers in scientific journals and international conferences proceedings. His area of interest is embedded systems and the reliability of complex systems