International Journal of Smart Home
Vol. 10, No. 3, (2016), pp.191-200
http://dx.doi.org/10.14257/ijsh.2016.10.3.19

An Embedded Software Power Consumption Model based on Software Architecture and Support Vector Machine Regression

Xiong Wei [1,2,*], Xiaobin Liu[1], Bing Guo[1], Shen Yan[3] and Wenli Zhang

[1]*Computer Science College, Sichuan University, Chengdu, SC 610064, China*
[2]*Leshan Vocational & Technical College, Leshan, SC 614000, China*
[3]*School of Control Engineering, Chengdu University of Information Technology, Chengdu SC 610225, China*
[1]*guobing@scu.edu.cn,* [2]*xwedu79@163.com*

Abstract

As embedded devices prevail in daily life, high energy consumption caused by embedded software caught academic attentions. Multifarious testing and predicting methods are developed accordingly. This paper proposes a model about energy consumption of embedded device based on analysis of embedded software structure and support vector machine regression. The nonlinear relationship between energy consumption and software structure is revealed. The research finds software structure is determined by features like number of components, complexity of component interface, component coupling, and path length. These features are qualified and modeled by using support vector machine regression and energy consumption is predicted based on this model. The experiments results confirm the proposed model.

1. Introduction

Embedded technology is used in many fields, such as industrial control, image processing [1] and even those used in people's daily life. The wide use of embedded devices raises issues of energy consumption. The annual power consumption of embedded devices has reached over l000 billion kw [2]. In embedded system design, energy consumption is also becoming a prominent consideration in the effort to prolong service time under the restriction of limited electricity. For example, in the literature [3-4], the author presents how to maximize real-time performance while consuming the least amount of energy. In the literature [5] the author utilizes an algorithm to reduce access time and energy consumption for scratch pad memories in embedded multicore systems.

Measuring and estimating the energy consumption for an embedded system has become a key issue, and much progress has been made in this regard. In the literature [6], the author puts forward an energy consumption model based on the (communicating sequential process) (CSP) algebraic language. In the literature [7], the author utilizes a time delay Petri net, modeling with tasks, and analyzes the relationships between tasks, communication protocol, etc. In the literature [8], the author proposes an N-linear function relationship in analyzing the software architecture characteristics in an effort to minimize energy consumption, but the accuracy of this model is poor in practice, so it cannot meet practical requirements. In the literature [9], the author puts forward a fitting and predicting model by using BP neural, but the defects of this model are neural network over-fitting and low generalization ability. Research on embedded system energy consumption, mainly comprises three levels: instruction level, software architecture structure level, algorithm level. The instruction level is shown in Figure 1.

Although analysis can only be done after the software is fully developed, estimating the energy consumption of embedded system is needed in the software design stage. Software architecture and algorithm level can provide energy analysis in the software design phase.

ISSN: 1975-4094 IJSH

algorithm level
architecture structure level
instruction level

Figure 1. Energy Consumption Analysis Levels

According to Hayes Roth's: software architecture is an abstract system specification, mainly including functional components, component interface and component connectors. As shown in Figure 2, components are encapsulations of a certain function, the connectors are the intermediary between components. Through interface, they interact with each other.

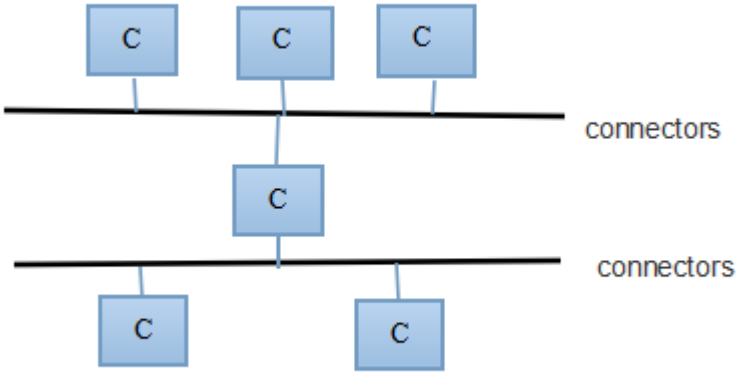


Figure 2. Software Architecture

In this paper, we consider software architecture structure elements as being closely related to energy consumption. Through experiments, we actually found that a few characteristics of the software system structure and the energy consumption have a particular internal relationship which cannot be simply regarded as a linear .or quadratic function. We extracted five characteristics which are closely associated with energy consumption, and used a support vector machine regression to fit and predict energy consumption and achieved favorable results.

This paper proposes an energy consumption model based on embedded software architecture characteristics and support vector machine regression. Section 2 describes the quantifying of the software architecture characteristics. Section 3 describes the support vector machine regression. Section 4 shows the process of the experiment. Section 5 concludes this paper.

2. Quantifying with Software Engineering Methods

Programs, (usually in c, c++, Java), are compiled into binary code, and hardware completes the corresponding actions, such as memory reading, writing and arithmetic logic operations, *etc.*, so as to accomplish a certain function. In the process, hardware executes instructions and triggers the corresponding circuit causing energy consumption.

According to the software architecture and the features of power consumption, this paper extracts five characteristics: valid code lines, number of components, complexity of component interface, component coupling, and path length. These characteristics can reflect how software driven hardware causes power consumption.

Definition 1: Embedded software power consumption equals the average power consumption multiplied by software running time, expressed as:

$$E = \int_{t1}^{t2} Pdt \quad (1)$$

E represents the total energy consumption, and t represents the running time. For accuracy, we test software energy consumption multible times and calculate the average as total energy consumption.

Definition 2: The average power consumption of embedded software is a nonlinear function of software structure characteristic (linear as a special category of nonlinear function). , expressed as:

$$P = f(C) = f(C1,C2,... Cn) \quad (2)$$

P represents the average power consumption;f represents a nonlinear function of embedded software characteristics;symbols C1, C2, . . . Cn represent software architecture characteristics.

2.1. Valid Code Lines Quantifying

Valid code lines are in the first consideration because it has a close relationship with the elements behind it. We divide program code into the categories of valid and invalid code. The main distinction is if there is a contribution to energy consumption.

Valid code is the code which can generate binary instructions by compilation. It mostly includes variable calculation, variable assignment, initialization functions, function calls, *etc*. In a fixed platform, each binary code has a fixed energy consumption value. And in a fixed compiler platform and environment, the generated binary code amount is also fixed. Therefore valid code lines affect energy consumption greatly. When calculating the valid code lines of a program, we consider the following situations:

a. for loop structure, unroll into basic assignment, computing statements are considered;
b. b. for virtual functions, only valid code lines of child class are considered;
c. for branch structure, according to the actual situations, the probability of each branch is set, multiplied by valid code lines of each branch, and the sum to count is calculated;
d. for child class, the public valid code lines of the parent class to count is added.

2.2. Number of Components

Components are an encapsulation of software modules accomplishing independent functions. The component number reflects the number of modules and valid code lines. So the number of components indirectly affects energy consumption. The software component number can be obtained by the design of the software architecture.

2.3. Average Complexity of Components Interface Quantifying

Through interface, components communication and accomplishing control and data flow interactions, component interface is divided into input interface and output interface. A larger interface number implies a greater chance of invocations of components. So component interface complexity has an indirect effect on energy consumption. According to the literature [10], the component j input interface complexity is expressed as the following formula:

$$I_j = m \times \sum_{i=1}^{q} e^{|Lj - Li - 1|} \quad (3)$$

The component j output interface complexity is:

$$O_j = m \times \sum_{i=1}^{q} 2^{|L_j - L_i + 1|} \tag{4}$$

The output and input interface complexity is:

$$A_j = \sqrt[3]{I_j + O_j} \tag{5}$$

So the average complexity of components interface can be expressed as:

$$R_C = \frac{1}{m} \sum_{j=1}^{m} A_j \tag{6}$$

(m---- software components quantity ; I_j ----- the input interface complexity;
q -----the total number of input interface ; Lj------ the layer depth of component j;
Li ------- the layer depth of component i.)

2.4. Path Length Quantifying

The number of components involved in running can be regarded as path length, therefore a longer path means more components involved in running. So path length has an indirect effect on energy consumption. Path length metrics has two types: single metrics and multiple-branches metrics. Sequential structure and cycle structure can be regarded as single metrics, and selective structure can be regarded as multiple-branch metrics. For single path, we can directly count the number of components as path length. As shown in Figure 3 (a), we can count five components for length path. For multiple-branches, firstly we multiply the component's number by the running probability of the branch, and secondly calculate the sum of all branches and take the sum as the path length. As shown in Figure 3 (b), if the path a-b-c probability is 30% and the a-b-d-e probability is 70%, we can get 3*30%+4*70%=3. 7 as the path length.

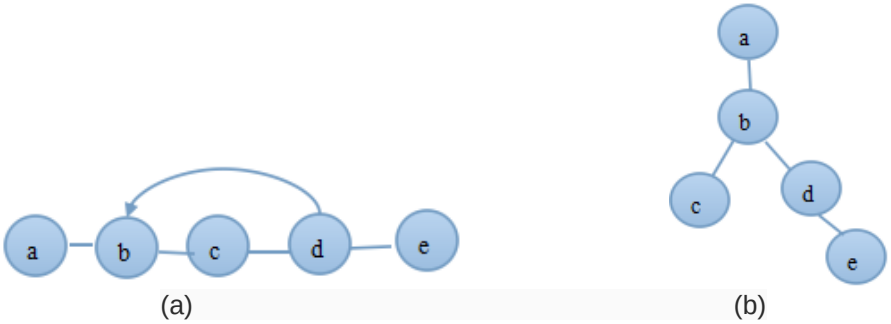


Figure 3. Path Length

2.5. Components Coupling Quantifying

Components coupling is the metric of correlation degree between modules. It includes control relationship, called relationship, and data transfer. The coupling strength depends on the component interface complexity. There are many mature methods of measuring coupling in software engineering, such as CK Metrics Suite and MOOD Metrics. This paper uses MOOD metrics. The average component coupling formula can be expressed as:

$$AC = \frac{\sum_{i=1}^{n} [\sum_{j=1}^{n} has\_coupling(C_i, C_j)}{TC^2 - TC} \tag{7}$$

Tc represents the total number of components.

$$has\_coupling(C_i, C_j) \begin{cases} = 1 & C_i \text{ and } C_j \text{ coupling} \\ = 0 & C_i \text{ and Cj no coupling} \end{cases} \tag{8}$$

3. Support Vector Machine Regression Prediction

Support vector machine is a pattern recognition and regression method based on statistical learning theory. It can successfully solve small samples, nonlinear and high dimensional pattern recognition problems.

Compared with artificial neural network and its applications [11], support vector machine is based on the principle of structural risk minimization, and its optimization goal includes two indicators: empirical risk and confidence risk. The neural network method is only based on empirical risk minimization, however empirical risk minimum is not equal to the actual risk being minimum. So the generalization ability and the small samples learning of the support vector machine are better than those of the neural network.

The support vector machine is used for classification and regression (SVR). For SVR, given a training set, the elements have an unknown distribution p (x, y):

$\boldsymbol{X} = \{(x_1, y_1), (x_{2,}, y_2)...(x_{n,}, y_n)\}$ with $x_i \in R^n$, $y_i \in R$,and the function

$$F = \{f \mid f : R^n \to R\} \tag{9}$$

The basic regression problem is to minimize the function

$$R[f] = \{\int c((y - f(x)), x) dP(x, y)\} . \tag{10}$$

In the above function, c is the loss function, and indicates how to penalize the errors between y and f (x).

Support vector machine regression (SVR) includes nu-SVR and ε-SVR [12, 13]. Their main difference is that the parameter nu is used in nu-SVR to control the number of support vectors, while the parameter epsilon is used in ε-SVR. In this paper we use ε-SVR. ε-SVR regression in trying to solve the following constrained quadratic programming problem:

$$\min \ \frac{1}{2}\|\omega\| + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{11}$$

$$\text{s.t} \begin{cases} f_i - y_i \le \varepsilon + \xi_i; \\ y_i - f_i \le \varepsilon + \xi_i^*, 1 \le i \le N; \\ \xi_i, \xi_i^* \ge 0 \end{cases}$$

Kernel functions are used in ε-SVR to map linear inseparable set in a low dimensional space to a higher dimensional space to separate through dot product operation.Kernel functions include:

linear kernel : $k(x_i, x_j) = x_i \cdot x_j$; (12)

polynomial kernel: $k(x_i, x_j) = ((x_i \cdot x_j) + 1)^d$ (13)

Radial Basis kernel : $k(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{\delta^2})$; (14)

sigmod kernel: $k(x, x_i) = \tanh(k(x_i, x_j) - \delta)$ (15)

Among them, the radial basis function (RBF) is widely used because compared with the other kernel functions, it has the following advantages:

a. the kernel function can map samples to a higher dimensional space;

b. linear kernel function is a special case of RBF;

c. compared with polynomial kernel function, the RBF requires fewer parameters.

In this paper, we choose RBF kernel function.

4. Experimental Results

In the experiment, we used HMSim and Libsvm. HMSim as a high precision instruction-level power consumption simulation platform, and simulated the single-core processor ARM7 instructions set. It supports Embedded Linux and UC/OSII operating system applications. We tested a program power consumption three times on the HMSim, and calculated the average as the program power consumption. We compared the test values with the predicted values of the programs and drew conclusions. All these procedures include Mp3 decoding, bubble sort, maze problem programs, *etc*. The experiments use Libsvm as ε-SVR regression tools. Libsvm is a simple and efficient SVM pattern recognition and regression software package. Using Libsvm for regression, there are three main steps: data normalization, parameter selection, training and prediction. , The selection of the parameters is critical, as it is directly related to the accuracy of prediction. The main process is as follows:

Step 1: data preprocessing and normalization

In order to improve training speed, and prevent the influence of large attribute values upon small attribute values, the original data has to be scaled to [1-1] interval.

Step 2: choosing kernel function and parameters

In this paper, we choose the RBF kernel function. ε-SVR regression using RBF kernel needs to optimize three parameters: gamma parameter of the kernel function (in Libsvm gamma=$1/2\sigma^2$) and the loss function 'c' and 'p'. The n-cross validation method is used to find the optimal parameters. In n-cross validation [14-15], the training set is divided evenly into subsets: $\ell 1, \ell 2, \ldots, \ell n$. At first, take $\ell 1$ as test set, $\ell 2, \ell 3, \ldots \ell n$ as training set. , The first group parameters are obtained,then take $\ell 2$ as test set, $\ell 1, \ell 3, \ldots \ell n$ as training set, get the second group parameters;. . . .; In this way get n groups of parameters, finally take the final average values as the parameters of ε-SVR. Using this method can prevent the occurrence of over fitting and enhance the generalization ability.

Step 3: Training and predicting

We use the obtained parameter values as training parameters to get the training model, and finally input the training model as parameters to predict the power consumption values.

The experiment has two groups: Group I and Group II, both including the training set and the test set. In Group I, the training set includes 11 procedures, and the test set includes 7 procedures. Through n-cross validation, we got the best training parameters c=1024, g=0. 25, p=0. 0009765625. In Group II, the training set includes 8 procedures, and the test set includes 6 procedures. Through cross validation, we got the best training parameters c=1024, g=0. 0625, p=1. 0. The following 4 tables are the training sets and test set. Tables 1 and 2 are the training set and the test set of Group I. Table 3 and Table 4 are the training set and the test set of Group II.

Table 1. The Training Set 1

SN	LOC	TC	Rc	Rp	AC
1	330	5	3.19156	11.5	0.35
2	421	7	2.8995	13	0.21429
3	467	7	2.66201	9	0.19048
4	452	7	2.66201	9	0.19048
5	710	8	2.78312	10.5	0.15625
6	319	4	2.96598	8	0.5
7	510	8	2.89715	13	0.2

8	526	8	2.98312	10	0.25
9	641	9	3.09651	12	0.14815
10	1437	11	3.10221	18	0.24825
11	1902	14	2.71982	21	0.20217

Table 2. The Test Set 1

SN	LOC	TC	Rc	Rp	AC
1	355	4	2.71982	6	0.41667
2	2027	22	4.00993	31	0.10451
3	301	6	2.27648	13	0.36111
4	468	6	2.71982	12	0.34012
5	1072	10	2.71044	21	0.24825
6	450	6	2.83525	12	0.23655
7	1100	11	2.79455	21	.0.36653

Table 3. The Training Set 2

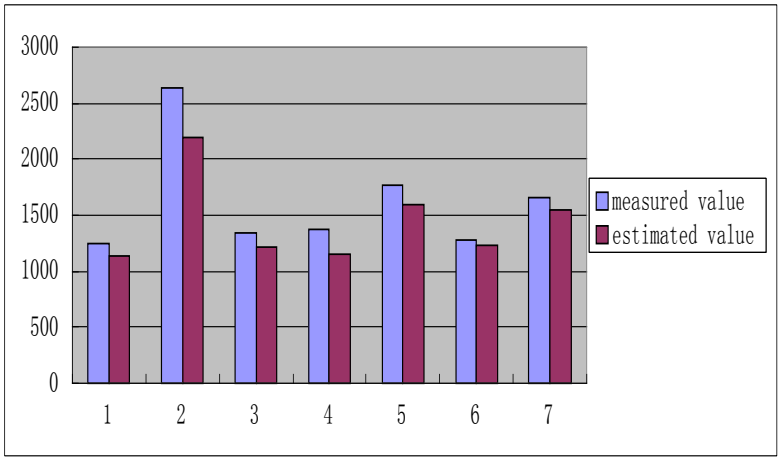
SN	LOC	TC	Rc	Rp	AC
1	82	3	2.04385	4	0.5
2	91	3	2.04385	4	0.5
3	284	4	2.96598	11	0.5
4	225	4	2.96598	8	0.5
5	172	4	2.71982	6	0.41667
6	149	4	2.71982	6	0.41667
7	440	5	3.2485	7	0.4
8	530	5	3.3455	7	0.4

Table 4. The Test Set 2

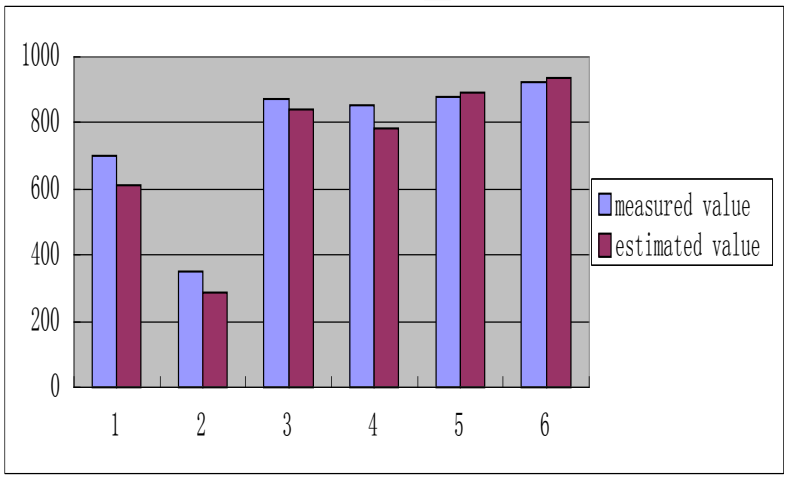
SN	LOC	TC	Rc	Rp	AC
1	197	5	2.27648	6	0.25
2	99	3	2.04385	4	0.5
3	212	5	3.19156	8	0.32
4	430	4	2.5785	7	0.4
5	450	5	2.6475	8	0.45
6	520	5	2.8435	8	0.5

(SN ------ the serial number of procedures; LOC------ valid code lines;
TC ------ the number of components; Rc------component interface complexity;
Rp------the average path length; AC------ the average coupling.)

The following figures are the results(comparison between measured and predicted values:



(a) The Group I Results



(b) The Group II Results

Figure.5. Comparison between Measured and Predicted Values

From the above figures, in Group I, the maximum devi-ation between the estimated values and the measured values is 17%, the minimum deviation is 4%. In Group II, the maximum deviation between the estimated values and the measured values is 19%, and the minimum deviation is 3%. The deviations are within the acceptable range. In summary,

1. Embedded software power consumption is significantly associated with software architecture characteristics, and their correlation being nonlinear is reasonable.

2. All five characteristics are closely related with energy consumption, but their impacts are different. Characteristics of valid code lines has the greatest impact among the five.

3. Using ε-SVR to fit and estimate the power consumption can effectively reflect the nonlinear relationship between energy consumption and software structure characteristics, and the small sample learning performance and generalization ability are preferable.

5. Conclusion and Future Work

This paper puts forward an energy consumption model of embedded software based on an embedded software architecture characteristic and support vector machine regression. The model reveals the nonlinear relationship between energy consumption and software architecture characteristic. According to software engineering methods, we firstly extracted and quantified five characteristics which are closely related to energy consumption, then used a support vector machine regression method to determine and estimate energy consumption. The experiments results are: the maximum deviation between the estimated

values and the measured values is 19%, and the minimum is 3%. The experiments proved the rationality of this thesis and effectiveness of the estimating method.

In future work, quantitative research on the correlation coefficient between software characteristics and energy consumption will be further researched. In addition, the relationship between other software characteristic and energy consumption will be studied.

Acknowledgment

This work was supported in part by the State Key Program of National Natural Science Foundation of China (No.61332001, 61272104 and 61073045).

References

[1] C. Wang and S. Zhu, "A design of FPGA-based system for image processing", Review of Computer Engineering Study, vol. 2, no. 1, **(2015)**, pp. 25-30.
[2] G. B; Y. Shen and Z. L. Shao, "The redefinition and some discussion of green computing Chinese Journal of Computers", no. 12, **(2009)**.
[3] J. Niu, C. Liu, Y. Gao and M. Qiu, "Energy Efficient Task Assignment with Guaranteed Probability Satisfying Timing Constraints for Embedded Systems", IEEE Transactions on Parallel and Distributed Computing (TPDC), vol. 25, no. 8, **(2014)** August, pp. 2043-2052.
[4] M. Qiu and E. H. M. Sha, "Cost Minimization while Satisfying Hard/Soft Timing Constraints for Heterogeneous Embedded Systems", ACM Transactions on Design Automation of Electronic Systems(TODAES), vol. 14, no. 2, Article 25, **(2009)**, pp. 1-30.
[5] Y. Guo, Q. Zhuge, J. Hu, J. Yu, M. Qiu and E. H.M. Sha, "Optimal Data Placement and Duplication for Embedded MultiCore Systems with Scratch Pad Memory", IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD), vol. 32, no. 6, **(2013)**, pp. 809-817.
[6] T. T. Zhang, X. Wu, L. CD and D. Yw, "On energy-consumption analysis and evaluation for component based embedded system with CSP", Chinese Journal of Computers, in Chinese with English abstract, vol. 32, no. 9, **(2009)**, pp. 1-8.
[7] L. Q. Chen, Z. Q. Shao and G. S. Fan, "Energy consumption modeling and analysis for distributed realtime and embedded systems", Journal of East China University of Science and Technology, (Nature Science Edition), (in Chinese with English abstract), vol. 35, no. 2, **(2009)**, pp. 250-255.
[8] C. Y. Seo, S. Malek and N. Medvidovic, "Estimating the energy consumption in pervasive Java-based systems In", Proc of the 7th Working IEEE/IFIP Conf. on Software Architecture(WICSA 2008) NewYork：IEEE Press, **(2008)**, pp. 277-280.
[9] L. XiaoBin and G. Bin, "Embedded Software Energy Modeling Method at Architecture Level", Journal of Software, vol. 23, no. 2, **(2012)**, pp. 230-239.
[10] X. M. Wang, "A model of measuring software structure complexity and its auto-realization", Computer Application, vol. 19, no. 6, **(2009)**, pp. 16-19.
[11] T. Wang and H. Yan, "Research fire alarm system based on extension neural network", Review of Computer Engineering Study, vol. 2, no. 1, **(2015)**, pp. 9-16.
[12] B. Scholkopf, A. Smola and R. Williamson, "New support vector algorithms", Neural Computation, no. 12, **(2000)**, pp. 1207-1245.
[13] B. Scholkopf, J. Platt and J. S. Taylor, "Estimating the support of a high dimensional distribution", Neural Computation, no. 13, **(2001)**, pp. 1443-1471.
[14] O. Chapelle and V. Vapnik, "Choosing multiple parameters for support vector machines", AT&T Research Labs, vol. 46, no. 3, **(2002)**, pp. 1-3.
[15] V Cherkassky and Y Ma, "Practical selection of VM parameters and noise estimation for SVM regression", Special Issue on SVM, vol. 24, no. 15, **(2002)**, pp. 82.

