822121104008:N.DHARSHINI
UNIVERSITY COLLEGE OF ENGIEERING
PATTUKKOTTAI

MEASURE ENERGY AND CONSUMPTION

PHASE-4 PROJECT SUBMISSION

Energy Consumption Measurement i

ABSTRACT

The low-power design of software and hardware is crucial for the efficiency and sustainability of embedded systems. Accurate energy consumption measurement plays a vital role in evaluating the energy performance of software and hardware, which also provides design insights to develop new paradigms for algorithm or code optimization in terms of energy efficiency and system cost. A few literature has surveyed this research field, but it lacks multi-faceted comparisons and comprehensive analysis. In this paper, we first show the necessity of accurate energy consumption analysis for embedded systems. Then, we study major methods in literature for measuring energy consumption of embedded systems, which can be summarized with three categories: 1) measurement-based energy profiling, 2) model-based energy estimation, and 3) simulator-based energy estimation. Some subcategories are further made based on characteristics of these apporaches. The pros and cons of each category have been reviewed and evaluated through multi-faceted comparisons. Finally, for transparent energy analysis and improving the energy efficiency of embedded systems, we come up with some contributing factors that matter the energy consumption measurements and discuss the challenges and future research directions.

INDEX TERMS

Internet of Things (IoT), embedded system, energy consumption, power consumption

I. INTRODUCTION

Battery-powered embedded systems play an important role in Internet of Things (IoT) architecture as the physical carrier, which covers from wireless sensors [1], [2], wearable devices [3], mobile smart devices [4], to ICT infrastructure equipments. With the achievement of IoT technology, the foreseeable growing trend of the total amount of global active IoT connections will reach 21.5 billion in 2025, as shown in Fig. 1, which was investigated by IoT Analytics [5]. Since an increasing number of devices are equipping with powerful abilities to handle computation-intensive tasks (e.g., machine learning training), the energy-constrained embedded devices are struggle to prolong working time. However, less knowledge of transparent energy usage throughout the embedded systems results in obstacles to design energy-efficient software and hardware [6]. Hence, accurate energy consumption measurement is imperative to optimize energy usage of embedded systems.

From the perspective of energy consumption sources, software instructs the hardware to perform operations such as calculation and memory allocation, while in this process the hardware consumes energy because of circuit switching. Consequently, the basic idea for acquiring energy consumption is measuring at hardware level. For example, some works estimate processor energy by modeling the energy consumption of CPU at transistor or switch level, where the switching power of transistors for computing is proportional to the capacitance of the transistor gates, the CPU switching frequency and working voltage, which is given in [7]. At a higher level, researchers model the power at logical gate level [8]. This type of modeling is more time-efficient but less accurate than which at the switch level. Others perform

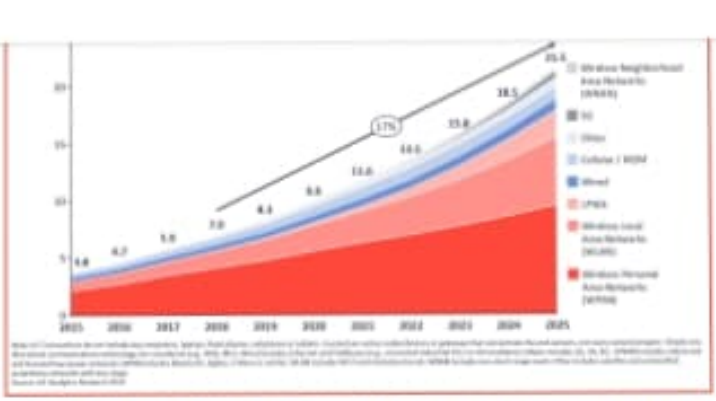

FIGURE 1. Global number of connected IoT devices1. It is predicted that the number of globally connected IoT devices will be 2.5 billion by 2025.

power analysis at higher hardware abstraction levels, such as register transfer level [9]. Further up is measuring or modeling at the level of hardware components, such as processors, memory, disks, and peripheral circuits, where many works have been done [10], [11].

On the other hand, as the driver of hardware executing, software instructions indirectly consume energy, which means the energy consumption can be mapped to software structure. The granularities of the mapping includes assembly instructions, functions, and software components [12], [13]. The study of energy consumption measurement of embedded systems provides traceable energy behavior information at various granularity levels, which is not only for evaluating the energy consumption of software/hardware, but also improve the developers' cognition of software energy characteristic for optimization. Besides, the energy measurement of embedded systems can be used to recognize rogue software since the malicious program execution can lead to additional energy consumption [14]–[16].

However, diverse research emphases and multi-levels energy consumption measurement schemes lead to the complexity of this field for researchers. We aim to provide a detailed analysis of the latest energy measurement techniques in the current background. Based on the analysis, we put forward the defects of the existing methods and future research directions. The main contributions of this work:

1) This paper proposes a novel and thematic taxonomy for classifying the existing works for energy consumption measurement. They are categorized into: a) methods of measuring and profiling, b) model-based energy estimation schemes, and c) simulator-based energy consumption estimation approaches.
2) We make a comprehensive comparison of these available methods in multiple metrics, including the power data source, measuring or modeling object, modeling methodology, granularity, and error rate. The capabilities and shortcomings within each category are analyzed.

The rest of this paper is organized as follows. Section III introduces the first kind of energy consumption measurement method, which is the combination of general instrument measurement and data analysis. The model-based energy consumption estimation schemes are described in Section IV. Section V presents software simulation-based energy consumption estimation approaches. Section VI discusses the challenges and future works in this field. And Section VII we concludes the paper.

II. THE TAXONOMY OF ENERGY MEASUREMENT METHODS

When investigating the existing literature about energy consumption measurement techniques of battery-powered embedded devices, it has been found that these available methods can be classified from multiple perspectives. For example, in literature [17], power modeling schemes of embedded systems are categorized into transistor or switch, logic gate, register transfer, and system level, according to different stages in design flow. To map energy estimation techniques to software applications, literature [18], [19] categorize the existing energy mesurment methods into code-analysis and mobile components power model based schems, which is based on the hierarchy of embedded systems. In terms of whether the energy consumption profiler is online or not, the existing methods could be classified into online and offline [20].

Through the analysis of above papers, it can be found that researchers choose the corresponding taxonomy according to the research purpose, and it is difficult to make comprehensive comparisons according to one single metric. Therefore, from the principle of measuring and modeling for embedded systems, this paper classifies existing methods into three categories: measurement-based energy profiling, model-based, and simulator-based energy estimation schemes, which is shown in Fig. 2. In the measurement-based energy consumption profiling methods, due to different tools for obtaining power data, there are two methods: instrument-based and platform-based. In the method of model-based energy estimation, many papers have obvious classifiable characteristics in the information types required for modeling. Therefore, such methods can be categorized into hardware utilization-based, system-call based, and program-analysis based modeling.

Secondly, after extensively investigate and summarize the characteristics of existing methods, we compare these techniques under each category with multiple metrics to achieve a comprehensive understanding. The comparison metrics we used include granularity, power data source, measuring o modeling object, modeling approach, overhead, and erro rate. The granularity refers to the level of energy consumption analysis. For example, the granularity at hardware leve includes transistor, logic, and component levels, and it a

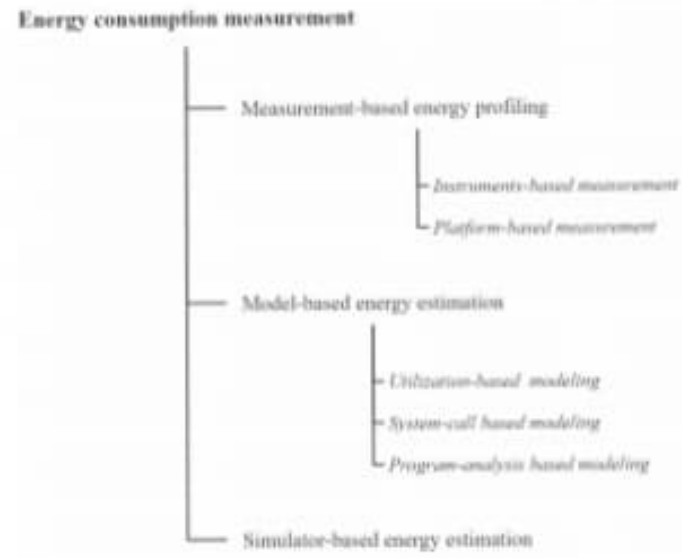

FIGURE 2. The category of energy consumption measurement methods.

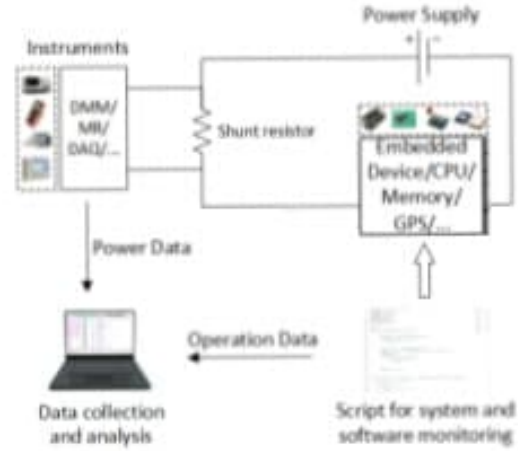

FIGURE 3. The basic framework of power measurement. A high precision shunt resistor is connected between the power supply module and the device. The probes of instruments (e.g., DMM, MR, DAQ) are attached to both sides of the resistor and sampling the voltage across it. The server obtains the corresponding data for energy consumption analysis.

the software level includes assembly instruction, source code line, process, function, and application levels. The power data source represents where the device power is acquired. The measurement or modeling object identifies the described targets for power consumption, including instruction, function, phone feature, component, and system. The modeling methods include linear and second-order polynomial regression model, piecewise constant model and finite state machine model. Based on these detailed comparisons, the pros and cons of existing methods can be summarized.

The more detailed analysis for each class of approach is given in Section III, IV and V, respectively.

III. MEASUREMENT-BASED ENERGY PROFILING

A. CATEGORY

1) Instrument-based measurement

In embedded systems, software drives hardware, and the energy is consumed in the operation process of the hardware circuit. Therefore, it is an intuitive method to measure the system energy consumption on the hardware circuit by using instruments.

Fig. 3 shows the basic framework of power measurement approach. Generally, a high precision shunt resistor is connected between the battery or power supply module and the device, which the device can be the whole embedded device or the hardware modules (e.g., CPU, memory and GPS module). The probes of instruments are attached to both sides of the resistor and sampling the voltage across it at a certain frequency. Consequently, real-time voltage can be acquired when the system is running, and the corresponding current and power consumption are calculated by Ohm's law and power formula. The measuring instruments include high precision Digital Multimeter (DMM), Memory Recorder (MR), Oscilloscope, and Data Acquisition (DAQ) card. At the same time, scripts are used to monitor the detailed execution procedures of software and operating system (OS) in the embedded system. Finally, the energy consumption can be cooperative analyzed based on the power and the software operating data.

The mainstream literature which adopts instrument-based methods are shown in Table 1, these studies measure the energy for the whole system. Then the energy consumption is attributed to corresponding applications according to the running period. For example, in literature [21], a method for measuring the energy consumption of mobile application program is introduced. The power trace of the device is characterized by the framework in Fig. 3. In [22], aiming at reducing energy consumption of the embedded devices which employ P2P protocols, an offline method is proposed for measuring the energy consumption. Similar to [21], high precision resistor with small resistance and DAQ card are used to collect voltage data at a fixed frequency. The application's operation information is recorded by accessing the specific GPIO port in the hardware device. Moreover, a device driver is developed to record threads and functions information for function-level energy consumption analysis, thereby the problem of energy consumption attribution could be solved.

In [23], a tool PowerScope was developed for estimating energy consumption of applications in mobile devices. As shown in Fig. 4, PowerScope is a typical framework that includes three software modules: the system monitor, the energy monitor, and the energy analyzer which are responsible for collecting system activity data, current data, and estimating energy consumption, respectively. An important feature of PowerScope is that it can map energy consumption to corresponding program structure, and this capability is dependent on symbol tables. However, energy consumption cannot analyze online since it is a post-processing tool. In practice, PowerScope needs a set of modifications to the kernel. The developers need to call a series of APIs and

FIGURE 4. The framework of PowerScope. The energy analyzer obtains PC/PID samples and current data from the system monitor and the energy monitor, respectively, and then combines the symbol table for energy consumption analysis.

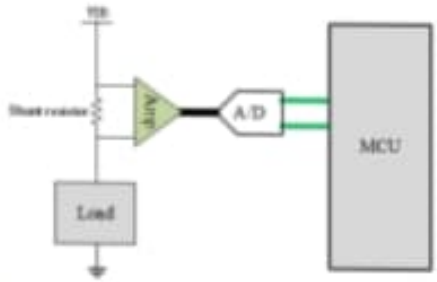

FIGURE 5. The typical system architecture of power measurement platform.

set corresponding parameters when they use PowerScope to estimate the energy cost for application, which will cause additional workload for developers.

Although the above methods can accurately measure the total energy consumption for overall mobile devices, the fine-grained power measurement is powerless. Due to hardware circuit conversion and other hardware call effects when application operation, it will unevenly attribute the consumed energy. To solve these issues, researchers adopt the approaches which use probes of instruments inserting the hardware circut to measure the interested component objects. For example, in [10], aiming at evaluating the energy usage of mobile devices, researchers proposed an energy measurement method for both the overall system and main hardware components, afterwards, energy models are built for the device under some usage scenario. In this method, a mobile device which can obtain circuit schematics is used because they need to set placeholders in power supply rails for each target hardware component. Then, sense resistors which with high precision and little resistance are inserted in these placeholders. DAQ card is used to collect the voltage drop data across these resistors directly. In this manner, energy consumption can be measured accurately.

Another component-level energy consumption analysis without modifying the hardware circuit is introduced in [24]. Aiming at improving the energy-efficiency for heart rate monitoring of smartwatches, researchers analyzed the energy consumption of specific hardware components such as acceleration sensor, screen, Bluetooth, and heart rate monitoring components. In this article, PCI-6230 is used to obtain the voltage drop over the shunt resistor at a certain frequency. They stress each component running separately for getting run-time power data, then the power consumption of each component is obtained by subtracting the energy when the watch is idle. However, DAQ cards or other instruments used in the above work are expensive, especially those with multiple probes. Also, in order to obtain the power data of hardware components, hardware circuit modification is needed, which is hard to implement for users.

2) Platform-based measurement

In addition to using general instruments to measure current and voltage, other researchers have designed power measurement platforms to replace instruments, and mainly to solve the problems with instruments such as expensive, large volume, and difficulty in flexibly establishing measurement experiments.

The typical system architecture of this kind of power measurement platform is shown in Fig. 5, a shunt resistor is put in series between the power supply and the load. Following a current sense amplifier or differential amplifier is used to amplify the small voltage drop across the shunt resistor and then provide it to the analog-to-digital converter (ADC). The ADC module is used to convert the analog signal into a digital signal and send it to the MCU, which is responsible for collecting the current data and sending it to the upper computer or directly processing it locally to obtain the load power consumption.

For instance, the researchers in [21] had developed a prototype board based on micro-controller used for measuring the current, but not used in the experiment in the paper. In [25], a testbed FlockLab was developed to reveal the system behavior into wireless embedded systems. The services provide by FlockLab include logic analysis, power profiling, adjusting power supply, and serial I/O logger. In terms of power profiling, it uses a small shunt resistor to sense the current density from battery to target device. A 24-bit delta-sigma ADC with sample rate of 28kHz is used to sample the voltage across the shunt resistor, which amplified by a current sense amplifier. PowerBench [26] is a testbed used to record the power traces of several sensor nodes in parallel. The current measurement method adopted in PowerBench is similar to that in [25]. However, its current measurement range is only from 0-60 mA. Meanwhile, the measurement range of FlockLab is from 2 μA to 160 mA, which is not suitable for high current devices so that limiting the scope of use.

Most of the previous works use a single shunt resistor's voltage drop to catch the current of the device, which leads to a small measurement range [25]–[27]. This kind of measurement platform is hard to meet the requirements of the modern embedded device since it may have high peak currents [28]. For that reason, some researches carried out to extend the measurement range of power consumption by adaptively adjusting the resistance of the shunt resistor. In Nemo [29], five resistors (0.1 Ohm, 1 Ohm, 10 Ohm, 100 Ohm, 470 Ohm) and four MOSFETs are used to compose a so-called

shunt resistor switch. The microcontroller can adjust the resistance dynamically according to the current intensity, so that ensure the measurement dynamic range from 0.8 μA to 202 mA. Similar to Nemo [29], Potch et al. [30] uses two shunt resistors (1 Ohm and 100 Ohm), comparator, and a MOSFET to compose a dual shunt resistor stage to replace a single shunt resistor, which enables the measurement range from 1 μA to several hundred mA.

However, although many studies above can measure precisely and have appropriate measuring range, their measurement objective is the whole mobile device, and that means they can accurately trace the mobile trace change over time, but cannot provide fine-grained analysis of energy consumption. Moreover, the circuit design of these platforms are relatively complex because many electronic components, such as resistors, op-amps, high-resolution ADCs, and microcontroller, are required. Meanwhile, these components will bring errors to the platform, and special attention should be paid to the device selection principles.

B. COMPARISON

Including the above research literature, many works use the methods which based on instruments or specially designed platforms to sample the power data, and apply data analysis technology to analyze the energy consumption of embedded systems. Table 1 compares the features of the works we reviewed under this category. The relative comparsion are analyzed as follows.

Since the purposes of energy measurement for embedded systems are different, the focuses of energy consumption analysis for existing work also varies. For example, literature [21]–[23] analyze the energy consumption at the software level with different granularity, such as protocol step, function, and process or procedure, while literature [10], [24] analyze the energy at hardware components level.

In most studies, external instruments are used for real-time voltage and current sampling. The complexity of hardware modification is relatively high, which is inconvenient for developers and users. Benefit by the high frequency and high-resolution instruments, high precision power data can be obtained in these experiments. But its shortcomings are that the setup of the hardware experiment environment is complicated, and the equipment is costly. Especially in [10], even the hardware circuit of the device needs to be modified. For this reason, some researchers developed hardware platforms for specific energy consumption measurement [25]–[27], [29], [30]. Compared to instruments, such platforms are cheap, small in size and can provide customized analysis based on the measurement objectives. However, due to the complexity and cost of circuit design, such a platform only has a few channels, a small measurement range, and difficult to provide fine-grained energy analysis.

Regarding the measurement error, all the literature studied utilizes offline methods for power data analysis, so the data processing on the Device under Tests (DuTs) or external computing devices does not result in additional operation

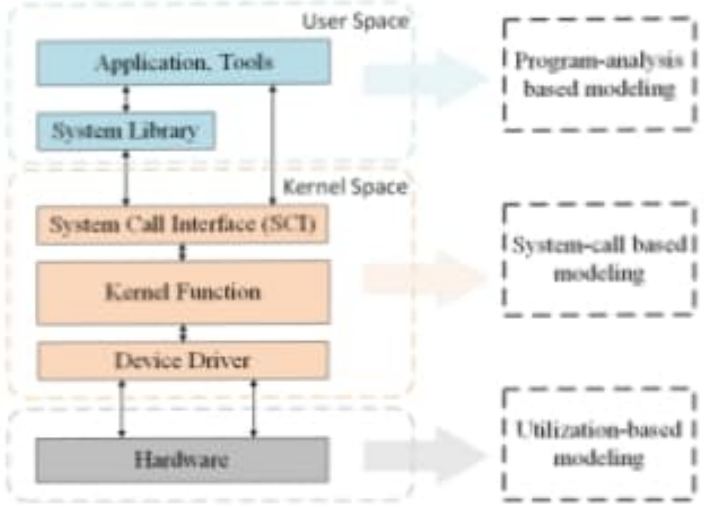

FIGURE 6. The modeling types corresponding to different levels of information in embedded system.

energy consumption. It is onl dul and application trace collection y c errors. Most of the researchers co e extra caused by this item is very small and can be neglect. addition, measurement errors may also come from the circuit settings. For the instrument-based schemes, the possible error source is the voltage drop of the shunt resistor. In the platform-based manners, error sources include not only this, but also from sampling frequency, voltage amplification, and analog-digital conversion.

IV. MODEL-BASED ENERGY ESTIMATION

Another scheme which has been studied widely is model-based energy consumption estimation. As shown in Fig. 6, embedded systems can be divided into three layers, include user space, kernel space, and physical hardware from top to bottom. During system operation, user-space applications cannot directly access or manipulate the hardware, but indirectly through the system call interface provided by the system kernel. In view of this hierarchical characteristic of embedded systems, model-based energy estimation schemes can be categorized according to different information sources. In the user space layer, software programs written in high-level or assembly language can be modeled by program analysis. At the kernel space level, system call data provides the access information of program to hardware resources, so it can be modeled based on the system call information. For the physical hardware layer, the energy consumption is reflected in hardware operation, so it can train the model based on the utilization rate of the hardware components.

A. CATEGORY

1) Utilization-based modeling

The correlation between hardware utilization and power consumption give rise to the idea of estimating the energy consumption of the system based on various hardware parameters, including CPU load, disk read and write accesses, and

Num	Granularity	Measuring Instrument	Measurement Object	Online/offline	Data Process Location
[21]	Protocol step	External (PCI-MIO-16E-4)	Voltage	Offline	Server
[22]	Function/task	External (DAQ)	Voltage	Offline	Local
[23]	Process/procedure	External (DMM)	Current & Voltage	Offline	Local
[10]	Hardware component	External (DAQ)	Voltage	Offline	Host machine
[24]	Hardware component	External (PCI-6230)	Voltage	Offline	External

TABLE 1. Comparison of instrument-based energy consumption measurement methods.

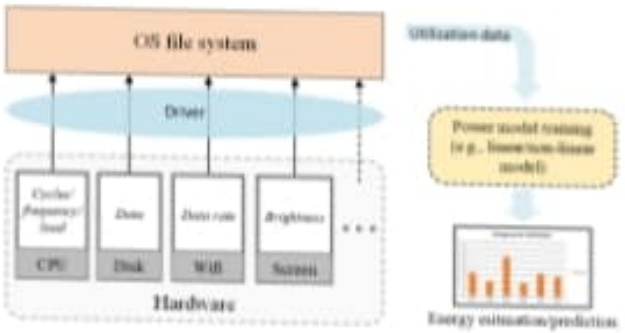

FIGURE 7. The utilization-based modeling. The main characteristic parameters of the hardware components reflect their utilization and are recorded by OS. Therefore, it is acquired from the OS file system for power modeling.

network transmits and receives, and the modeling process is shown in Fig. 7. The main characteristic parameters of the hardware components reflect their utilization. Therefore, researchers acquire these utilization data from the OS file system for power modeling.

For instance, in literature [11], a system-level energy consumption estimation model based on hardware components utilization was proposed. Researchers considered the users activity drives the execution on mobile devices, which sequentially determines the power consumption. User usage patterns can be represented by the combinations of hardware units and their utilizations, such as CPU utilization, screen brightness, and the count of bytes transferred with Wifi module. The hardware components utilization is as model parameters and acquired from the hardware performance counters (HPCs). Under this consideration, a logger application for Android G1 mobile phone is developed for logging system performance metrics, and a piece of external equipment is used to sample the current data in the application running process. The researchers use linear regression to build the model. The power model for a certain component is:

$$p = a * \beta,$$

where p is the power consumption, and β is the parameter which can affect the power state switches and a is the corresponding coefficient.

Then, the power consumption P of the whole system is:

$$P = k + \sum_{i=1}^{n} p_i = k + \sum_{i=1}^{n} a_i * \beta_i \qquad (1)$$

where k is constant, which indicating the power that not attribute to any measurement, and n is the number of main hardware components. After the power consumption model is trained for each main hardware component, it can accurately estimate the power consumption of any application and provide the energy breakdown for hardware units. However, defects still exist. For example, in the modeling process, they need to design a series of tasks to stress different components for each hardware component of the mobile devices. Whether the training task design is reasonable or not may have a grate impact on the accuracy of the model. Secondly, this model just suit certain use scenarios, which means when mobile devices are used in different environments, it may no longer be applicable.

Other literature also adopts linear regression to model the power consumption of hardware components [31], different from [11], they aim at detecting battery exhaustion attacks on mobile devices. The detection system consists of power estimation module and process identify module which responsible for establishing liner model and identify process-by-process power consumption, respectively. The test results show that the average estimation error is 5.67%. In order to manage the power consumption, literature [32] proposed a system-level power modeling method based on linear regression analysis. Different from the previous work [11], [31], which modeling energy consumption with general linear regression, the model coefficients in this article are restricted to non-negative since the energy consumption of the device is non-subtractive and it also makes coefficient matrix sparse [33].

In addition to using linear regression models, paper [34] introduces a second order polynomial model to approximate the transmit energy usage of WiMAX mobile devices which running IEEE 802.16e protocol. The transmit energy consumption per bit is determined by transmission power, data rate, and file size. The power consumption model has guiding significance for the construction of energy-effiecient wireless sensor network.

Most of the previous energy modeling methods for embedded systems need external devices for auxiliary measurements [11], [31], [32], such type of manual modeling methods not only make the modeling process time-consuming and laborious but also cannot extend self-adaptively to other embedded devices. To solve this problem, literature [35]–[38] proposed automatic liner regression power modeling methods using battery discharge behavior. For example,

PowerBooter [35] uses built-in battery voltage sensor and the battery discharge behavior to monitor the power consumption of the main components in the device. There is a certain relationship between the battery voltage and state of discharge (SOD).

The power P from time t_1 to t_2 is calculated by:

$$P * (t_1 - t_2) = E * (SOD(V_1) - SOD(V_2)), \quad (2)$$

where E is rate capacity of the battery, $SOD(V_i)$ represent the state of discharge under battery discharge voltage V_i. After the power model is built, system-level energy consumption can be estimated online using the developed PowerTour tool. For improving the accuracy of the liner regression model, principal component analysis (PCA) is used in Sesame [36] to transform the model parameters such as CPU and memory usage data. However, some components which are invisible to the operating system does not take into account because the parameter data is collected from the operating system (*sys* and *proc* file system).

Similarly, built-in battery monitor unit (BMU) is used to monitor the power usage with battery interface for adaptive modeling in [37]. The CPU and network card usage statistics which from hardware performence monitors and software performance counters are used as model parameters. For the applications which include both computation and communication parts, they should be divided into subtasks and distributed to different processors. Then, the corresponding model is chosen for matching. However, this run-time model is built for application which only uses computation and communication resource and cannot provide a comprehensive energy estimation for the great majority interactive applications.

In literature [38], an energy consumption modeling method based on SOD is proposed for embedded mobile devices. The authors indicate that different hardware components in the same power supply system may affect each other, which implys that in the case of a combination of device states, it is not accurate to calculate energy consumption by referring hardware spec sheet. The system-level energy consumption model of the device in interval T is $E_a(n) = P_a(S_a(n)) * T + A_a(n)$, where $A_a(n)$ is the adjustment parameters indicating mutual influence between components, and $P_a(S_a(n))$ is the power in devices' operating state S_a. It has been proved that the energy pridiction from this model is more accurate than using only the spec sheet published by the manufacturer. However, although the state effects between components are considered, this model cannot provide fine-grained energy attribution.

Linear regression models are built in the above utilization-based methods, while piecewise constant models were used in the following literature to simplify the power model of phone feature. For example, in order to minimize energy consumption and optimize robustness when using location-based service on mobile devices, the piecewise constant model is built for each phone feature [39]. In another work [40], an unsupervised power profiling system PowerProf is designed for estimating and predicting API-level energy consumption for applications on mobile devices. In this article, the continuous power are discreted to power states, and the power states of each phone feature are restricted to four and separated by running time t of applications. Then, the genetic algorithm is applied to find the optimal power state and corresponding separation time t. After modeling, tests show that high accuracy can achieve. The advantage of PowerProf is that it is implemented in the application layer, without modification of the operating system and kernel. Besides, no human-device interaction is required in the training process. However, the system-level power consumption estimation is hard to attain. Only part of phone features that providing location-based services are modeled by PowerProf, and lack of modeling for other components.

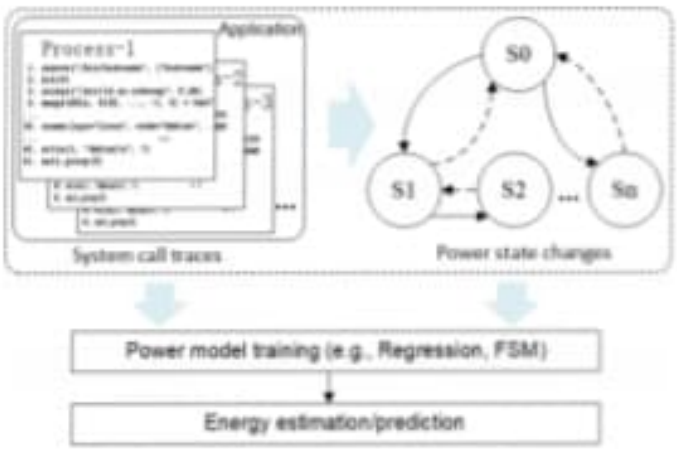

FIGURE 8. The system-call based modeling. Processes in the application have different system call trajectories, which lead to power state changes. This information is used to train the model.

2) System-call based modeling

In embedded OS, the system call interface provides the only way for software programs to manipulate hardware, which means it will lead to power state changes when processes in applications call interfaces, as shown in Fig. 8. Thus the mapping between system calls and power state changes of embedded systems is established in some studies. This mapping information is used for training the power model. The benefits of system-call based modeling approaches are reflected in three aspects. First, it provides the only way for applications to access hardware components; second, such models can capture all power consumption behaviors of system calls without component workload or utilization, which is depicted in Fig. 9; third, it naturally related to calling subroutines, threads, and processes. The modeling process is shown in Fig. 8.

In [41], researchers point out that the utilization-based power modeling approaches are inaccurate when modeling the hardware components in modern smartphones since there are several power behaviors not directly correlates the component utilization, such as tail power states of several components (e.g. GPS and SD card) and exotic components

FIGURE 9. The comparsion between utilization and system-call based modeling.

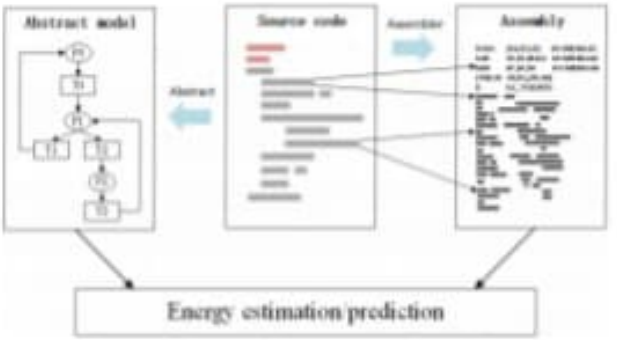

FIGURE 10. The program-analysis based modeling. The source code analysis is abstracted as a component model, while the assembly code is built to instruction-level energy model.

(e.g. GPS and camera). In order to solve this problem, they proposed a system-call based power modeling scheme which use the Finite State Machines (FSM) to model the power states and the state transitions of each component as well as the whole device. However, limitations are still exists in the modeling process. Typically, only up to three power states are adopted to represent the different running states when constructing FSMs for each component. For example, only two fixed States, base state and high CPU state are used to represent the states of CPU, while the process of their changes between these two states is ignored. Besides, in practice, external environments may lead to several power consumption which different from that measured in the laboratory or the datasheet. Therefore, it is not accurate to use the fixed state to model the running state in different using environments.

Based on previous work [41], Pathak et al. developed a fine-grained energy profiler Eprof for smartphone applications in [42]. Eprof uses the online FSM [41] to model the power consumption of smartphone components. Test results on different applications show that the error of Eprof is less than 6% at the process level granularity, while that of utilization-based and split-time approaches is 3% - 5% and 15% - 80%, respectively. However, the researchers also mentioned that the FSM power model cannot be used on high rate components because of the high overhead for modeling.

In [43], researchers proposed AppScope, which is an energy measurement framework for Android smartphones and implemented as a kernel module in Linux. In AppScope, an system-call based linear regression model is adopted to approximate the power consumption of hardware components in smartphones. This linear model is similar to which proposed in [11], but the non-utilization-based information is used for modeling. Furthermore, in the model training phase, instead of using external equipment to measure the current or voltage between battery supply and device, DevScope [44]–an application that is used to monitor power consumption for each component by assigning workload to it for training–is used to collect the power data. However, the test results show that the estimation error can be relatively large in some case, and the reason is the model is too simple with regards to CPU, which ignore the effects of cache and bus.

Inspired by system call traces are used [41], [43], [44] for estimating resource usage, and combined technique from Green Mining [45], an energy prediction model, GreenOracle [46], is proposed for estimating the energy consumption of Android applications. Before modeling, it needs to collect large amounts of data (e.g., the counts of different system calls and CPU utilization rate) of different applications as features, then, big data approaches are used to train the model. In this work, four machine learning algorithms: ridge regression, lasso, support vector regression, and bagging were employed for energy consumption modeling. System call times, CPU utilization, and other relevant information data ware used as feature parameters to train the model. The advantage of this method is that it reduces the complexity of using hardware measurement instruments and make it convenient for developers.

3) Program-analysis based modeling

The thought of program-analysis based modeling approach is shown in Fig. 10. Source code can be abstracted upwards or compiled into assembly or machine code. The models based on program analysis in userspace are categorized into assembly instruction level model and software component level model according to different stages and granularity of software.

a: Assembly instruction level model

In computer systems, the user-space software are compiled into binary instructions, which are then executed one-by-one by the processor. Therefore, many works focused on studying instruction-level power consumption modeling. Generally, the power of a single instruction is modeled by executing n (n tends to infinity) cycles to obtain the total energy and then take average. The total energy of software is the summation of all instructions' energy consumption. For example, as shown in Fig. 11, the embedded security algorithm AES128 is executed on Beagle Bone Black, and the corresponding instruction stream is recorded and analyzed. Consequently, instruction-level and function-level energy consumption are calculated.

In this category, Tiwari et al. put forward the energy consumption analysis at the instruction-level for the first time. In literature [12], the modeling approach was proposed based on external measurement and instruction analysis. Although

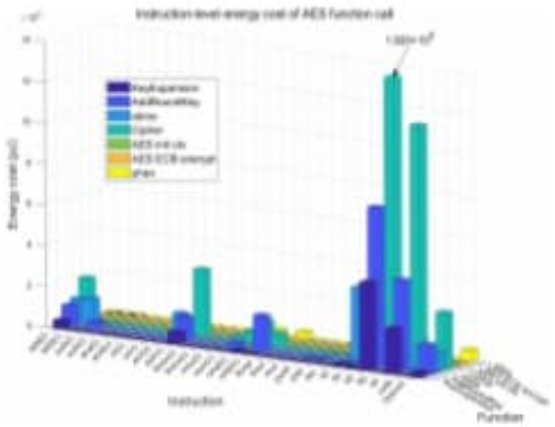

FIGURE 11. **Perform assembly instruction-level energy consumption analysis on the operation of the AES128 algorithm to encrypt 32 bytes string on Beagle Bone Black board.**

the basic energy consumption of the software program is the summation of energy consumed by each instruction, there existing inter-instruction effects, which include 1) effect of circuit states; 2) effect of resource constraints and cache misses. Thus the energy consumption of these effects should be taken into account. Then, the total power consumption model is built according to the base costs of the instruction set and the cost brings from the inter-instruction effects. Furthermore, a method of hypothetical test based on the constant model is performed [47].

Based on the previous work [12], an instruction-level power analysis method for specific embedded DSP software has been proposed in [48]. This new modeling method considers the special architectural feature of the DSP processor which allows instructions to be packed into pairs and spends less energy than separated. Another energy overhead is from the non-adjacent multiply instructions, and the effect is solved by adding an average overhead penalty to base cost. In the literature, the modeling for a specific DSP processor architecture achieves high accuracy but makes this method less scalable.

Similar to [12], instruction-level modeling approach has been proposed in [49]. The total energy includes the energy consumption of processor core, flash memory, memory controller and SRAM. The model parameters include instruction operation code, shift operation times, register group bit flip, instruction weight, and Hamming distance, and different types of memory access.

Per instruction power model is also used in literature [50], in which Hao et al. proposed an energy estimation tool Elens for applications. Elens consists of Software Environment Energy Profile (SEEP) and program analysis. Specifically, for each different hardware component, SEEP uses a function to estimate the energy cost of each instruction in each power state of the hardware component. Program analysis module records each path traversed through each method of the application by the code instrumentation technique in [51]. Then, different granularity such as path, method, and source line-level energy consumption can be estimated. It has been proved that the accuracy of this method reaches 10%.

b: Software component level model

Due to the most existing methods need complete program source code for energy consumption evaluation, it is difficult to carry out energy consumption analysis in the early stages of software development. So the researchers of literature [13] proposed to use the probability branchs and iterations in program flowchart to evaluate its energy consumption in the early stage of program development. Energy consumption analysis of main elements of flowchart includes processing flow (addition, subtraction, multiplication, and division.), decision flow (conditional statement, and circular statement.), input and output, and so forth. However, many energy consumption elements are not represented in the flowchart, such as data storage and background process, so the model needs to be modified. The energy model is $E_{esti\_program} = E_{flow} * \tau_{corr}$, where E_{flow} is determined by the basic element energy consumption of flowchart, statement branching probability p, and statement execution number n. The execution energy consumption of the basic elements in the flow chart is obtained by measuring the current during the cycle execution by external equipment and calculating the average value. The parameters p and n are determined by the developer, therefore, the error sources in the model are mainly correction parameter τ, branch probability p, and cycle number n.

B. COMPARISON

For the works of energy consumption estimation based on modeling, we make comparison in granularity, modeling object, modeling method, methodology, physical quantity measurement internal or external, online or offline, overhead and error, and the details are shown in Table 2.

The existing literature models the energy feature from software and hardware. At the software level, the modeling object can be instruction, function, task, or system [31], [32], [34], [36], [37], [40], so that the energy consumption granularities include instruction, process, application or system level. Generally, at the hardware level, the modeling object is hardware components [11], [35], [39], [43], [49]. The granularity of energy consumption analysis can reach component, API, routine, process or system level. In terms of modeling granularity and energy consumption granularity, fine-grained analysis can be realized no matter from the hardware or software level.

In model-based energy consumption estimation for embedded systems, the linear regression model has been widely accepted [11], [31], [32], [35]–[37], [43]. For the sake of reducing the size of the regression parameter set, [32] further used a linear model with non-negative coefficients. These regression schemes require parameters that have a linear relationship with the modeled object, and the most important is that parameters need to be easy to obtain. Compared with the higher-order regression model, it has fast convergence, better generalization performance, and energy-saving characteristics.

Num	Granularity	Modeling Object	Modeling Method	Measuring manner	Methodology	Online/ Offline	Overhead	Error
[12]	Instruction	Instruction	Constant model	External	N/A	N/A	N/A	Within 3%
[39]	Phone feature	Phone feature	Piecewise constant model	Internal	N/A	Online	N/A	N/A
[40]	API	Phone feature (Hardware component)	Conditional constant model	Internal (Smart battery interface)	N/A	Online	N/A	0.143 for median error
[50]	Application/ method/ source code line	Instruction	Constant/ linear function	External	Program analysis	N/A	0.2%–7.2%	Within 10%
[11]	Component	Component	Linear regression	External	Utilization-based	Offline	N/A	6.6% for median absolute relative error
[43]	Component	Component	Linear regression	External	System-call based	Online	35mW	Within 14.7%
[31]	Process	System	Linear regression	External	Utilization-based	Online	low	5.67%
[34]	Bit	Transmit function	Second-order polynomial regression	External	N/A	N/A	N/A	N/A
[32]	System	System	Liner regression with non-negative coefficients	External	Utilization-based	Online	N/A	2.62% median error
[35]	System	Component	Linear regression	Internal (Sensor)	Utilization-based	Online	80mW	Within 2.6%
[36]	System	System	Linear regression	Internal (Smart battery interface/ fuel gauge)	Utilization-based		1%	5% for 1s interval
[37]	Application	Task	Linear regression	Internal (Battery monitor unit)	Feedback-based (utilization)	Online	low	1% for CPU bound/ task; 6.6% for network task.
[41]	Routine/ thread/ process	Component	Finite state machine	External	System-call based	Online	5.4%–9.8%	0.2%–3.6%

TABLE 2. Comparison of model-based energy consumption measurement methods.

From the perspective of power measuring equipment, the methods of adopting external instruments and internal devices account for half of each. For the external style, these data acquisition instruments usually sample at a high rate and acquire high-precision power data. However, it is necessary to connect the instrument probes between the power supply and the device or even the hardware component (a schematic of the device is required), which leads to the high complexity of the hardware configuration and high cost. In addition, the power data collected by the instrument is difficult to send to the data analysis server in real-time, resulting in data analysis, model establishment, and update only in offline mode. That is suitable for measuring the energy consumption of embedded devices deployed in a static environment. As for the internal devices (e.g., fuel gauge, BMU, current/voltage sensor), they cannot sample too fast because of the extra power consumption caused by themselves and the demand for control. Besides, these internal devices usually introduce unavoidable errors due to analog-to-digital conversion. Based upon the interaction of all these factors, internal devices are generally hard to achieve the accuracy of the external instruments. On the other hand, the advantage of internal device measurement is that developers can obtain the power data by calling the interface at any time, and adjust the sampling frequency according to the demand. This enables online modeling and automatic model updating possible, and the ordinary users to evaluate energy consumption measurement freely.

From the perspective of methodology, there are mainly two modeling bases, namely utilization-based and non-utilization-based methods. The utilization-based schemes [11], [31], [32], [35], [36] are based on the usage statistics of components to reflect their power state changes. For example, the widely used technology of dynamic voltage

and frequency scaling (DVFS) dynamically adjusts the power supply voltage and the running frequency of the CPU according to the task demand, which leads to different levels of power states. However, in today's embedded system design, to achieve the purpose of power optimization, the intelligent terminal uses different levels of power management design [52], [53] in the system and driver, resulting in the poor accuracy of this power model in utilization-based power estimation. Another popular non-utilization-based method [37], [41], [43] is event-driven and system-call based. System-call based models capture the events that have power consumption but without utilization rate such as tail power state [54] of hardware components, which makes the error rate of fine-grained energy consumption estimation results far lower than that of utilization-based approaches. However, it should be noted that not all the system calls can be mapped to source code if source code level power consumption judgment need to be performed.

In the early stage of software energy consumption model development, the majority of works focus on instruction [12], [40], [47], [48]. Although the approach is intuitive, the modeling process has complicated steps and only for the energy consumption of processor and memory, which is not enough for other parts and the whole embedded device. For this consideration, some works employ the constant parameter model to approximate the phone feature and hardware component power characteristics, further to the entire system [39], [49]. Another widely studied method is to use linear regression to model the energy consumption of hardware components [11], [31]–[37], [43]. The relationship between power states and the main parameters of most hardware components enable linear regression analysis. In addition, there are two other methods for modeling, such as the program flowchart-based method and system-call based method. The program flowchart-based method [13] enables developers to roughly estimate the application energy cost in the early stage of development, which conducive to develop energy-efficient applications. Above variety models make the energy consumption estimation of embedded systems more convenient, but they may also cause errors. For example, when the environment in which the device is modeled is completely different from the environment used. Besides, after a period of use of embedded devices, the battery aging is inevitable and means the output voltage will lower than the rated voltage. However, in most experiments in the literature, direct current (DC) constant voltage source were used to replace the battery, which introduced errors.

V. SIMULATOR-BASED ENERGY ESTIMATION

Except for the above two kinds of approaches, another energy estimation scheme for embedded systems based on simulation software has received continuous attention. As shown in Fig. 12, the simulator consists of guest OS and basic power models (e.g., instruction-level model). For embedded energy consumption estimation, the simulator first presets electrical parameters of the hardware devices and

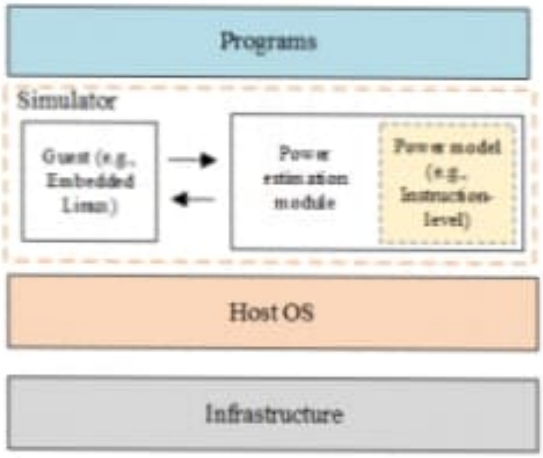

FIGURE 12. The framework of simulator-based modeling.

simulates the embedded OS so that programs can run on it. The software operation information is collected by the power evaluation module and then combine power model to estimate the energy consumption.

Based on the instruction-level power model [12], the idea of building the corresponding simulator was proposed by [55]. In [56], power consumption simulator Wattch for microprocessors was proposed, which has been a widely used energy consumption simulation tool. In Wattch, according to the hardware architecture, processor units is categorized into four lower-level categories: array structure, fully associative content addressable memories, combinatorial logic and wires, and clocking. The energy consumption characteristics of these units is represented by a parametric model and finally compose the total processor simulator. After that, Austin et al. developed an instruction-level simulation tool, SimpleScalar toolset, in literature [57], which has intergrated Wattch into it and is accurate to the cycle level, and it has been used in embedded system research. SimpleScalar is able to simulate the energy consumption when executable code running on embedded systems. Besides, the number of clock cycles, power consumption and other important information occupied by the code in the whole execution process are quantized and output.

In [58], based on SimpleScalar, Mudge et al. developed a set of energy consumption simulator Sim-Panalyzer which is suitable for the most widely used embedded Arm microprocessor. The Sim-Panalyzer modeling the power consumption in various aspects of the processor, including cache, data path and execution block, and I/O. While the Sim-Panalyzer can achieve energy estimates for some user applications, it cannot support the simulation under the OS environment, which is a more complicated case. In order to solve this problem, reform the existing simulator and increase the support for the OS environment are needed. In [59], based on Sim-Panalyzer, researchers extended the power consumption simulation to embedded Linux OSs. The problem of mixed energy consumption caused by OS and concurrent tasks is solved by using the task energy sheet and function energy

stack mechanism.

Another energy simulation framework EMSIM [60] is presented for analyzing the energy consumption of embedded Linux system which running on the StrongARM processor. In EMSIM, microprocessor (includes an instruction set, cache, and memory management unit (MMU)) and various system function components (e.g., interupt controller, timers, UARTs and memory) are modeled in sufficient detail to enable the operating system could execute on it. The instruction-level energy model of the StrongARM is directly obtained from [61], which includes cache, MMU, clock generator, timer, and internet controller. The energy comsumption of other components such as memory and UARTs are modeled by the data from data sheet of Itsy pocket computer [62]. Afterwards, a task and function energy accounting mechanism is built. The experiments show that the energy consumption estimated by simulating is generally lower than the actual measurements. In some cases, the errors are even greater than 20%.

A. COMPARISON

In order to realize the power simulation of processors and other components, most of the simulation software uses the instruction-level model. Simulators estimate the energy consumption of embedded software rapidly, and enable researchers and developers of embedded systems to obtain the required energy consumption data without setting up a hardware environment. These capabilities shorten the development cost and time, and lay the foundation for power consumption analysis and optimizing for embedded software. Unfortunately, the development and verification of simulation software usually takes a long time. When developing simulators for embedded systems, the first step is to simulate various functions of the operating system to support applications. Second, the energy consumption of the application is calculated based on the datasheet or the existing power models. As a result, the simulator usually only supports some classical embedded OSs and hardware components, which ignores factors in the actual circuit. In summary, the simulator is helpful to program development difficult to achieve accurate energy consumption estimates.

VI. CHALLENGES AND FUTURE WORKS

The potential and advantages of low power consumption of hardware will not be fully exploited without the rational utilization and efficient optimization of software. The premise of energy consumption optimization of software in embedded systems is accurate, fast, and low-cost local energy consumption measurement, which has been paid attention in recent years. From the hardware-level to software-level as well as the combination of both, the energy measurement techniques of embedded systems has developed rapidly. Specifically, significant progress has been made by the energy profiling of measurement-based approaches, model-based estimation schemes, and simulator-based estimation methods. The results of energy measurements are developing towards fine-grained, low overhead, and high accuracy. On this basis, some issues to be studied include the following aspects, and the directions need to be studied are shown in Fig. 13.

FIGURE 13. Research directions in energy consumption measurement.

a: Energy consumption measurement in fine granularity

Fine-grained energy measurements consist of fine-grained power data acquisition and energy profiling. The existing energy consumption measurement techniques mostly measures the overall energy consumption state of electronic equipment, which is not specific. Only a few works realize the runtime power measurement of several hardware components of embedded devices by using instruments at the hardware level. However, the energy consumption measurement only supports main hardware components such as the processor and memory and lacks the supporting for other hardware modules such as GPS and wireless communication module. On the other hand, most of the existing energy profiling methods are at the hardware component, function, or process level, which are inferior to fine-grained energy profiling schemes such as at source code line level and path-level because the laters are more helpful to developers. Thus the fine-grained power data acquisition and energy profiling are both need to be promoted.

b: Energy consumption measurement in an online fashion

When the external instrument is used to detect the power consumption of the device, only offline energy consumption detection function is supported, that is, after the power consumption data collection task, the instrument will upload the power data to the upper computer for analysis. This post-processing method makes it difficult for developers to monitor the energy consumption of the equipment in real-time and to achieve the purpose of online monitoring.

c: Energy consumption measurement in parallel paradigm

The number of channels of general instruments is limited, and the physical probe is usually unable to go deep into the hardware circuit for measurement. In addition, the method based on the power consumption data from datasheet also has data accuracy problems in different application scenarios. Thus these problems make the system unable to carry out

multipoint and parallel energy consumption measurement. The main reason for the above problems is that there is no real design and manufacture of a special platform for measuring the energy consumption of application software of embedded devices. Only approximate test data or theoretical derivation are used to evaluate the energy consumption of application program, which can only reflect the rough trend and cannot provide accurate energy consumption analysis.

d: Dynamic measurement of battery energy consumption

Embedded devices are usually powered by lithium-ion batteries, and there are many nonlinear behaviors in the battery charge and discharge process because of the electrochemical reaction, such as current effect, recovery effect, and temperature effect. These nonlinear behaviors of the battery have a high uncertainty effect on the endurance time of embedded systems. However, the most existing test platform uses a DC power supply which cannot directly measure the dynamic energy consumption behavior of the application program on the battery-powered embedded device and its corresponding battery capacity change characteristics. Only a few studies according to battery discharge behavior to analyze energy consumption.

e: Energy consumption measurement in an automated and unsupervised way

Most of the existing energy consumption measurement methods need to customize for the embedded platform, such as connecting the instrument to the target device, designing and developing specific software tasks for model training, and selecting model parameters. However, these procedures make the process of energy consumption measurement complex and highly dependent on experienced developers. Therefore, it is necessary to develop new methods that can automatically measure or unsupervised modeling for developers and users.

VII. CONCLUSION

With the rapid commercial deployment of 5G, the establishment of large-scale Internet of Things is possible, and the number of battery-powered embedded devices as IoT terminals will grow rapidly, too. As a result, the optimization of energy consumption of the battery-powered equipment becomes more and more urgent, and the research of fine-grained energy consumption measurement of embedded systems as an auxiliary tool to establish energy consumption ratings for different applications and help programmers to develop energy-saving applications becomes increasingly important. In this article, we have investigated several existing studies in this field. According to the methods for energy consumption measurement or estimation, they are divided into three sorts: energy consumption measurement based on instrument measurement and data analysis, model-based and simulation-based energy consumption estimation. These three kinds of methods have pros and cons in different situations, which have been analyzed above in detail. In each method sort, we compare the energy consumption measurement methods in multi-dimension within specific literature. After summarizing the characteristics of the existing methods, this paper points out the deficiencies of them and the future research directions: fine-grained problems, parallel and online measurement problems, automatic and unsupervised measurement problems and so forth. At the same time, these challenging problems are also urgent to be solved by researchers.

ACKNOWLEDGMENT

This work was partially supported by the National Key Research and Development Program of China under grant 2020YFB2104300, the S&T Major Project of Inner Mongolia Autonomous Region in China under grant 2020ZD0018, the National Natural Science Foundation of China (NSFC) Key Project Program under grant 61932014, the NSFC SINO-GERMAN mobility programe under grant No. M-0132, the State Key Laboratory Program under grant SKLD20Z02, and the National Development and Reform Commission of China (NDRC) under grant "5G Network Enabled Intelligent Medicine and Emergency Rescue System for Giant Cities".