applied sciences

MDPI

Article

An Energy-Efficient Strategy for Microcontrollers

Huanjie Wu, Chun Chen and Kai Weng *

College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China; 3100100294@zju.edu.cn (H.W.); chenc@zju.edu.cn (C.C.)
* Correspondence: wengkai@zju.edu.cn; Tel.: +86-571-8795-3955

Abstract: Power saving has always been an important research direction in the field of microcontrollers. Dozens of low power technologies have been proposed to achieve the goal of reducing their power consumption. However, most of them focus mostly on lowering the consumption rate. It is well known that energy is the integral of power over time. Thus, our view is that both power and time should be carefully considered to achieve better energy efficiency. We reviewed some commonly used low power technologies and proposed our assumptions and strategy for improving energy efficiency. A series of test sets are designed to validate our hypotheses for improving energy efficiency. The experimental results suggest that time has no less impact on energy consumption than power. To support the operation of the processor, some peripheral components consume a constant amount of power regardless of the clock frequency, but the power consumption will be reduced when the processor enters low-power modes. This results in some interesting phenomena that are different from the usual thinking that energy can be saved by increasing processor clock frequency. For STM32F407 and Xtensa LX6 processors, this article also analyzes and calculates the minimum sleep time required for achieving energy saving based on our analytical models. Our energy efficiency strategy has been verified, and in some cases, it can indeed improve energy efficiency. We also proposed some suggestions on hardware design and software development for better energy efficiency.

Keywords: embedded systems; energy consumption; energy efficiency; internet of things; low power technology

Citation: Wu, H.; Chen, C.; Weng, K. An Energy-Efficient Strategy for Microcontrollers. *Appl. Sci.* **2021**, *11*, 2581. https://doi.org/10.3390/app11062581

Academic Editor: Alexandre Carvalho

Received: 7 February 2021
Accepted: 9 March 2021
Published: 14 March 2021

1. Introduction

With the popularization of the concept of green computing [1], people are paying more attention to energy efficiency in computing [2]. Nowadays, large quantities of embedded systems are deployed everywhere in our daily life to satisfy the demands of modern smart life. Unlike general-purpose computing devices, embedded systems are often required to satisfy the constraint of low energy consumption. Both the operator and the users expect these devices to last longer and require little maintenance. Energy consumption has a significant impact on battery life, thermal design, device stability, and security. Additionally, energy consumption is an essential index of the performance of embedded systems. Microcontrollers provide computing and processing capability for embedded systems and have to follow energy constraints as well. Adegbija et al. [3] pointed out that IoT processors must be optimized for energy efficiency and configurable to achieve optimal execution, especially in terms of energy efficiency. Dozens of studies about low power technology have been performed by other researchers and companies, and to reduce power consumption, many low power technologies have been proposed as well. However, because energy consumption is the integral of power over time, power is not the only impact factor in energy consumption. Although low power technology contributes greatly to reducing the power consumption of embedded microcontrollers, it also affects the performance and execution time of applications. In the field of power consumption modeling, Sun et al. [4], Martinez et al. [5] Yoon et al. [6], and Alawnah et al. [7] also mentioned that especially for mobile smart devices and IoT devices, the periods of the

operating time have a non-negligible impact on the energy consumption of embedded systems. Existing studies focused more on the impact of time overhead introduced by low-power techniques on system performance. We believe that the energy overhead of low-power techniques and the effect of time on energy consumption should be taken into consideration and analyzed when drawing up energy-saving strategies.

In this paper, we reviewed some commonly used low power technologies and methods and proposed two assumptions about energy consumption. First, since energy is the integral of power over time, in some cases, energy might be saved even if instant power consumption increases. Second, the overhead introduced by processor clock frequency transitions and power mode transitions must be considered when applying dynamic voltage and frequency scaling (DVFS) and power modes. Several analytical models and a series of test sets are designed and conducted to validate our assumptions. The test results show that our assumptions are correct, and in some cases, our ideas can indeed contribute to improving energy efficiency. Based on the experiment data, we also propose several suggestions on hardware design and software development for better energy efficiency.

Contributions: This research presents some contributions in the domain of improving energy efficiency of microcontrollers.

(1) Analytical models are designed to analyze the effect and overhead of different low power technologies.
(2) A series of tests is conducted, and the power consumption of essential peripheral components shows its impact on energy consumption.
(3) A method to compare and choose the optimal among different low-power technologies is proposed to apply low-power technology more appropriately and effectively.

The paper is organized as follows. Section 1 introduces the background issues of energy efficiency in embedded systems and microcontrollers. Section 2 presents the existing low-power technologies for embedded systems and microcontrollers, points out their shortcomings, and presents our ideas. Section 3 contains our analytical models corresponding to our assumptions. Section 4 contains the measuring methodology and experimental data used in the study. Section 5 analyzes and discusses the results obtained from the experiments, as well as the recommended energy efficient strategies in different scenarios. Sections 6 and 7 provide suggestions for future improvement and directions and summarize the research.

2. Related Studies and Motivation

In this section, we first investigate existing low-power technologies and methods and then raise our assumption on improving energy efficiency.

2.1. Related Work

We have searched multiple combinations of keywords such as ‘embedded system AND low power’, ‘embedded system energy saving’, ‘embedded system extend battery life’, and ‘embedded system AND energy efficient’ to retrieve the peer-reviewed articles of journals, conference proceedings, book chapters, and reports from the databases of ACM Digital Library, IEEE Xplore, Engineering Village, and Web of Science. We also used Google Scholar as a complementary search engine to find other related documents as well as those not formally published. On the topic of energy-saving and extending battery life in the field of embedded systems, research can be roughly divided into two directions: increasing the energy harvest and decreasing the consumption.

Energy harvesting is a promising technique that takes advantage of environmental or other energy sources such as solar, wind, thermal gradients, and radiofrequency radiation. Chandrakasan et al. [8] stated that the ultimate goal of micro-power systems is to power microsystem devices using energy harvesting techniques such as vibration-to-electric conversion or through wireless power transmission. Gollakota et al. [9] pointed out that, with the development of computers, it is possible to power small computing devices using only incident RF signals. Gudan et al. [10] presented a system for measuring ambient

RF energy in the 2.4 GHz ISM band and suggested there is enough energy to support a low duty cycle wireless sensor node system. Li et al. [11] proposed a solar energy driven, multi-core architecture power management scheme called SolarCore, which is capable of achieving the optimal operation condition of solar panels autonomously under various environmental conditions with a high green energy utilization of 82% on average. Raghunathan et al. [12] presented the design, implementation, and performance evaluation of Heliomote, a prototype to address the key issues and tradeoffs arisen in the design of solar energy harvesting and wireless embedded systems.

Dynamic voltage and frequency scaling (DVFS), power mode management (also known as dynamic power management), and software layer low-power optimizing techniques are the most widely used power-saving approaches. Dynamic processor frequency is widely used as a power-saving method to decrease power consumption. Hua et al. [13] studied the optimal number and values of voltage levels for achieving energy efficiency with minimal area and power overhead of voltage regulators and of voltage transitions. Gheorghita et al. [14] proposed a technique for saving energy in embedded systems by using a smaller voltage at a time when less computation power is required.

Choi et al. [15] proposed a DVFS technique that enables one to achieve a precise energy-performance tradeoff while making use of runtime information about the external memory access statistics and to choose the optimal CPU clock frequency and the corresponding minimum voltage level based on the ratio of the on-chip computation time to the off-chip access time. Saewong et al. [16] proposed four DVFS techniques for saving energy in embedded systems: using a single frequency for the entire execution, using slack to save extra energy in low-priority tasks, which are suitable for systems where the overhead of DVFS is very high, using different optimal frequencies for every task, and minimizing the energy consumption based on monitoring actual execution times; however, these are unsuitable for on-line use due to their high complexity. Quan et al. [17] propose two DVFS algorithms for saving energy in real-time embedded systems; the first algorithm finds the minimum constant speed, and the processor is shut down when idle, and the second algorithm produces both a constant speed and a schedule of variable voltages for minimizing the energy. Kan et al. [18] proposed a DVFS technique for saving energy in soft real-time embedded systems by choosing the available frequency that is closest to the optimal frequency for a task.

In embedded SoCs, there usually exist several operating modes or power modes that can be used to save energy. Different modes require different electric currents to work with and take different periods of time to return to the normal mode. Generally, modes that consume less energy will take a longer time to return to normal mode [19]. A large amount of research has been performed to find the balance of energy-savings and real-time performance. Li et al. [20] proposed a method for selecting power modes for the optimal power management of embedded systems under timing and power constraints. Hoeller et al. [21] proposed an interface for the power management of hardware and software components. Huang et al. [22] propose an energy-savings technique that works by adaptively controlling the power mode of the embedded system according to the historical arrival of tasks. Bhatti et al. [23] present an online framework to integrate DVFS with a power-mode management (PMM) scheme to save energy in embedded systems. Niu et al. [24] propose a technique to save both leakage and dynamic energy in embedded systems by integrating DVFS and PPM. In 2013, Liang et al. [25] proposed a method to save energy in the user equipment, whereby the user equipment switches to sleep mode during their nonactivity periods and wakes up when required. They also discussed a strategy to prolong the sleep period of the sensors for better energy efficiency.

Ahmad et al. [26] assumed that different ARM instructions have similar power consumption and proposed a framework for static-analysis-based smartphone application energy estimation. Chen [27] conducted a test on ARM926EJS among different types of data, operators, and function types and obtained a result that achieved the same function using different instructions that could have different energy consumption. In addition, modern

compilers such as the GNU C Compiler (GCC) [28] and ARM Compiler (ARMCC) [29] also provide optimization options, whereby the compiler can attempt to optimize for small code size and high performance.

However, lowering power consumption does not necessarily mean saving energy. Chung et al. [30] researched the relationship between energy consumption and the program executing efficiency and pointed out that, in some situations, the growth rate of the power consumption is not always as high as that of the program execution efficiency. Thus, in this situation, the energy consumption can be lowered by improving the program execution efficiency. Dzhagaryan et al. [31] researched the relationship between the processor frequency and the energy consumed by program execution and pointed out that, with an increment in the processor frequency, the power consumed by the processor is increased, while the execution time of a program is reduced. The energy consumed by program execution can increase or decrease, varying based on the processor frequency. They proposed that the most energy-efficient processor clock frequency is below the designed maximum frequency, and the relationship between the energy consumption and frequency presents a U-shaped curve. In addition, because the voltage transitions can require time on the order of tens of microseconds [32], Jiangwei et al. [33], Pinheiro et al. [34], and Kuehn et al. [35] pointed out that the operation of processor frequency scaling can also introduce additional energy consumption because of the transition time overhead.

These studies have taken the overhead introduced by applying low-power technologies into account, but they focused more on designing low-power strategies from the perspective of system performance, especially system capacity and real-time performance. We believe that the overhead is not only a problem of system performance and energy consumption; it also has impacts on the DVFS strategy because it could be fairly significant and cannot be ignored. In addition, we want to figure out a convenient way to help determine when and which low-power technique has the best energy efficiency under specified circumstances.

2.2. Our Assumptions

As mentioned above, many low power technologies have been proposed to reduce the power consumption of microcontrollers. However, it is known that energy is the integral of power over time, and both the power consumed, and the execution time will affect the total energy consumption, illustrated by Equation (1) below:

$$E = \int_{t=0}^{t_{execution}} P(t)dt, \quad (1)$$

Reducing power consumption is not the only way to save energy; saving time should also contribute to energy saving. Our first assumption was that a higher processor frequency leads to more power consumption but a shorter execution time, so in the case of computationally intensive tasks, a higher processor clock frequency may not necessarily mean more energy consumption. In [32–35], it was pointed out that some widely used low-power technologies such as DVFS and power modes may have heavy overheads of transitions. We want to figure out a way to determine the application scenarios of different low-power technologies, so that microcontrollers can always run in the most energy-efficient way.

Since the use of peripherals is inevitable in real applications, the energy consumption of the embedded system can be divided into two parts: the energy consumed by the processor and the energy consumed by the peripherals. To verify our assumptions and understand the impact of low power technologies on energy consumption, it is essential to propose corresponding analytical models for each low power technology and analyze their energy consumption with actual experimental results.

3. Analytical Models

In this section, we propose analytical models that correspond to our assumptions. With the help of these models, it is possible to validate and quantify the impact of different low-power technologies on energy efficiency. To validate our assumptions and determine the overhead, we decided to take a deep look at energy-savings technologies bottom up. The whole work is divided into two parts: 1. the relationship between energy consumption and the processor clock frequency; 2. the overhead introduced by the dynamic processor frequency or DVFS and the overhead of applying power saving modes.

3.1. How Will Processor Clock Frequency Affect Energy Consumption?

The processor frequency has a significant impact on both the working current and the code execution time. It is known that the power consumption will increase as the processor clock frequency increases, and for computationally intensive tasks, the execution time will be shortened by increasing the processor clock frequency. Except for programs that contain delays, almost all of the programs can speed up the execution by increasing the processor frequency, and in ideal situations, the execution time and processor clock frequency will show an inverse proportional relationship. The STM32F407 used as a target device has a wide range of processor clock frequency up to 168 MHz. The question of "how [processor clock frequency will] affect energy consumption" can be translated into whether Equation (2) holds, where E_{af} represents the energy consumption at the adjusted frequency, and E_{of} represents the energy consumption at the original frequency. Energy is the time integral of the power, or the product of power and time if the power is almost constant. We can expand Equation (2) into Equation (3), where P_{af} represents the power consumption at the adjusted frequency, t_{af} represents the execution time at the adjusted frequency, P_{of} represents the power consumption at the original frequency, and t_{of} represents the execution time at the original frequency. To make the comparison more intuitive, Equation (4) is derived. Figure 1 is a schematic diagram of the energy consumption of the application code before and after adjusting the processor clock frequency.

$$E_{af} < E_{of} \tag{2}$$

$$P_{af} \times t_{af} < P_{of} \times t_{of} \tag{3}$$

$$P_{af} / P_{of} < t_{of} / t_{af} \tag{4}$$

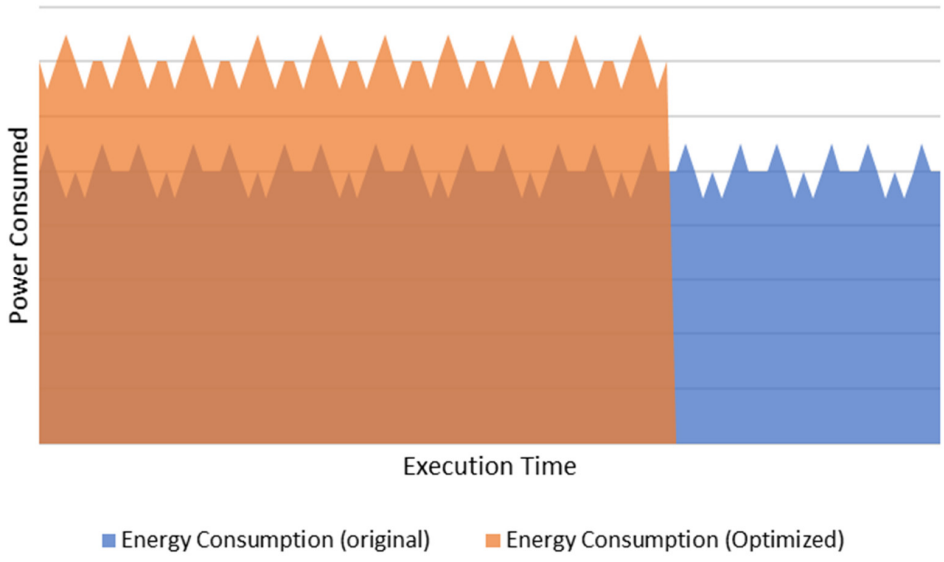


Figure 1. Schematic diagram of the energy consumption before and after adjusting the processor clock frequency.

The area formed by the power consumed during the execution is the energy consumption of the program. The blue area is E_{of}, the energy consumption of the original program, and the orange area is E_{af}, the energy consumption of the optimized program. If the orange area is smaller than the blue area, then it will be possible to save energy by adjusting processor frequency. This criterion means that the increment in the power consumption must be less than the speedup of the execution time.

However, even in a core board, there are many peripheral components that serve the system and program. These on-board peripheral components, other than the processor, also consume energy, while the whole system runs in active mode. Therefore, for Equation (3), in addition to the power consumption of the microcontroller itself, P_{af} and P_{of} also include the power consumed by other peripheral components. If the power consumption of the peripheral components is separated from the power consumption of the entire system, we can obtain the following:

$$P_{of} = P_{ofp} + P_{ofpc} \tag{5}$$

$$P_{af} = P_{afp} + P_{afpc} \tag{6}$$

P_{ofp} and P_{afp} stand for the power consumption of the microcontroller itself at the original clock frequency and the adjusted clock frequency, respectively, and P_{ofpc} and P_{afpc} stand for the power consumption of other peripheral components at the original and adjusted processor clock frequency. Equation (3) can be derived to

$$(P_{afp} + P_{afpc}) \times t_{af} < (P_{ofp} + P_{ofpc}) \times t_{of} \tag{7}$$

$$P_{afp} \times t_{af} + P_{afpc} \times t_{af} < P_{ofp} \times t_{of} - P_{ofpc} \times t_{of} \tag{8}$$

$$P_{af} \times t_{af} - P_{of} \times t_{of} < P_{ofpc} \times t_{of} - P_{afpc} \times t_{af} \tag{9}$$

In this situation, energy will be saved as long as the total energy consumption of the processor and peripheral components at the adjusted frequency is less than that at the original frequency.

3.2. How Will Overhead Affect Energy Consumption?

DVFS and power-saving modes are commonly used low power technologies. The STM32F407 used as the target device supports the runtime processor frequency transitions. By modifying Phase-Locked Loop (PLL) parameters and setting PLL as the clock source, the clock frequency of the STM32F407 processor can be up to 168 MHz. During the transition, the high-speed internal crystal oscillator (HSI) or high-speed external crystal oscillator (HSE) is used as the clock source, and now PLL parameters can be modified. After setting the PLL parameters, the PLL output is enabled, and when it is ready, the clock source is switched to PLL output. If peripherals are required, SysTick, the peripheral clock, and all the peripherals need to get reinitialized as well. It can be seen that adjusting the processor clock frequency may take a lot of time, so the impact of the overhead of processor clock frequency transitions on the energy consumption should be calculated.

In addition, the STM32F407 processor also comes with three power-saving modes: sleep mode, stop mode, and standby mode. It is evident that entering and waking up from power-saving modes will introduce energy consumption overhead, and the overhead should be calculated as well. Utilizing DVFS technology and applying power-saving modes have many similarities: they change the power consumption of the microcontroller, and both of them introduce a certain amount of overhead. The impact of their overhead on energy consumption can be studied in a similar way. Whether applying low power technologies such as DVFS and power-saving modes will contribute to energy saving can be expressed as Equation (10), where E_{lp} represents the energy consumption after low power technologies applied, E_{oh} represents the overhead energy consumption of applying low power technologies, and E_{ori} represents the energy consumption under the original condition. In Equation (11), the energy consumption is expressed by the product of power and time. Thus, we derive Equation (12), and the key to this question is whether the energy

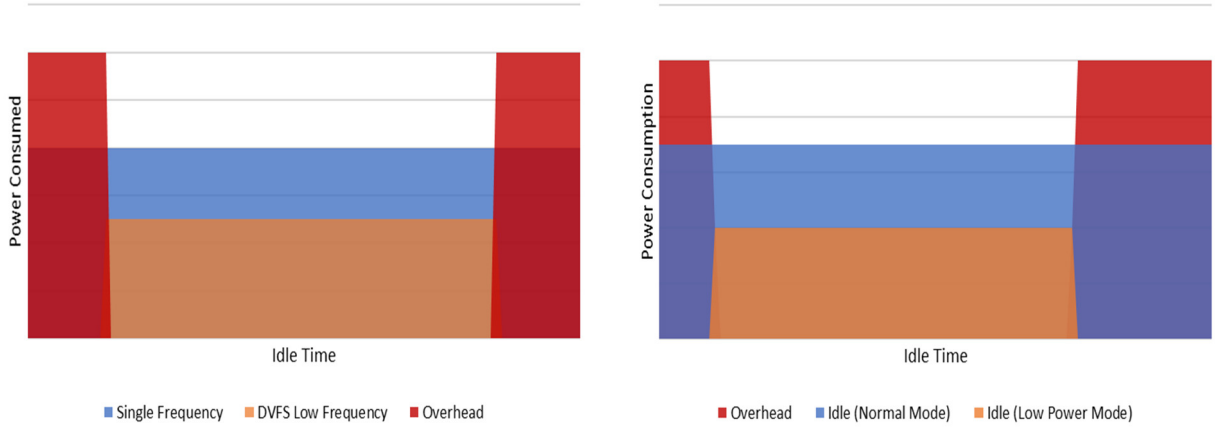
saved by applying low power technologies can cover the overhead that is introduced. Figure 2 shows schematic diagrams of the energy consumption of applying DVFS and power-saving modes.

$$E_{lp} + E_{oh} < E_{ori} \tag{10}$$

$$P_{lp} \times t_{lp} + P_{oh} \times t_{oh} < P_{ori} \times t_{ori} \tag{11}$$

$$P_{oh} \times t_{oh} < P_{ori} \times t_{ori} - P_{lp} \times t_{lp} \tag{12}$$

The total energy consumption of an embedded system after low power technologies being applied is made up of two parts: E_{oh}, the overhead (the red area), and E_{lp}, the energy consumption of the system (the orange area). Energy can be saved when the sum of the red and orange area is smaller than the blue area, E_{ori}, the energy consumption under the original condition. The overhead of DVFS in (a) includes two processor clock frequency transitions. The first transition is to adjust the frequency to the target frequency, and the second is to reset the frequency to the previous frequency. However, the overhead of applying a power-saving mode in (b) is slightly different from that of DVFS: the energy consumed to enter the power-saving mode and the energy consumed to wake up. There will be at most one system clock reconfiguration when waking up.



(**a**) (**b**)

Figure 2. (**a**) Schematic diagram of the program execution energy consumption at a single frequency and DVFS. (**b**) Schematic diagram of the system idle energy consumption at a single frequency and using power-saving modes.

If the overhead introduced by applying low power technologies is significant enough, we must consider whether the energy saved by using these technologies is sufficient to cover their overheads. In this case, a minimum time for which the system is required to stay in power-saving modes is

$$t_{min} = \frac{E_{oh}}{P_{ori} - P_{oh}} \tag{13}$$

For two different power-saving modes A and B, if Mode A has a better power-saving effect and less overhead, Mode A should always be the best choice. However, if the overhead of Mode A is higher than that of Mode B, the minimum time for which applying Mode A will have a better energy-saving effect is

$$t_{min} = \frac{E_{ohA} - E_{ohB}}{P_{lpB} - P_{lpA}} \tag{14}$$

4. Experiments and Results

In this study, an STM32F407 core board was used as the target hardware. It is designed by STMicroelectronics, and made in China. To validate our assumptions and ideas, a series

of test sets was designed to demonstrate the relationship between energy consumption and time. The experiments were performed on the STM32F407 core board to study energy consumption behaviors. In this section, we will introduce our measuring methodology, present our measuring results, and analyze the measuring results to show the ability to improve energy efficiency with different methods.

4.1. Measuring Methodology

In our previous papers [36,37], a design of the energy consumption measuring platform for embedded systems was proposed. In this platform, a KEITHLY 2280S-32-6 type high precision digital power supply was used to provide power consumption data, which is a product of Tektronix. The sampling rate of the KEITHLY 2280S-32-6 type high precision digital power supply can reach 25 kHz when the measuring resolution is set to $3^1/_2$ digits, and the sampling rate is 5 kHz when the resolution is $4^1/_2$ digits [38]. The digital power supply was equipped with General Purpose Interface Bus (GPIB or IEEE 488 bus), Ethernet, and Universal Serial Bus (USB) interfaces, and it supports queries using commands such as Standard Commands for Programmable Instruments (SCPI); as a result, the real-time power consumption data of the core board can be easily obtained. Its output voltage is set to 5.000 V in this study.

STM32F407 is a widely used microcontroller based on ARM® Cortex®-M4, and its operating frequency can reach up to 168 MHz. Its working current is approximately 238 μA/MHz, according to STMicroelectronics's website [39]. The ESP32 module is a low-power, feature-rich MCU with integrated Wi-Fi and Bluetooth connectivity for a wide range of applications developed by Espressif-System and manufactured by Taiwan Semiconductor Manufacturing Company, Limited (TSMC). It is equipped with an Xtensa® 32-bit LX6 processor. In this study, an STM32F407 core board and an ESP32 module were used as the target devices. The core board was equipped with the STM32F407 processor, the lead-out pins, and essential peripheral components that support the basic functionalities of the board. Figures 3 and 4 show the setup and connection of our measuring platform.

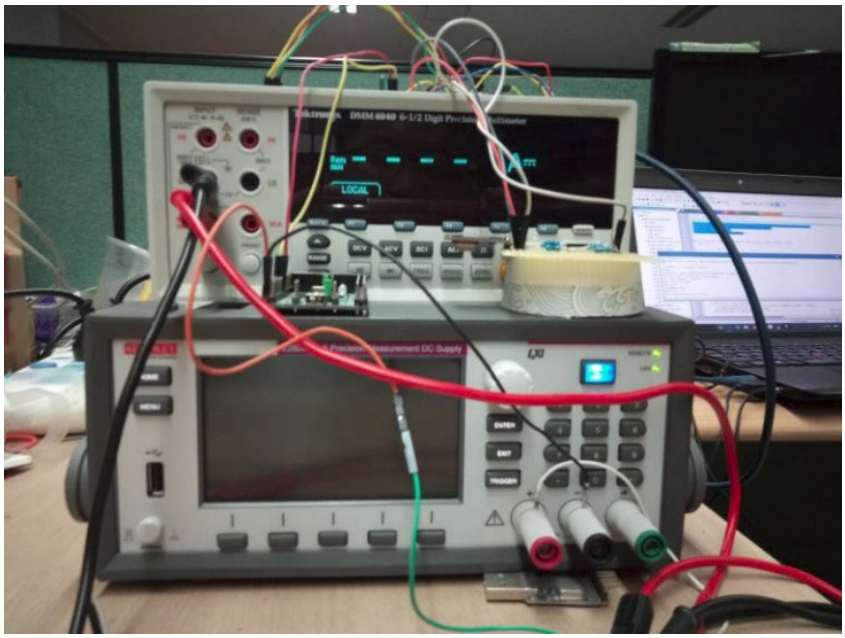


Figure 3. The measuring platform.

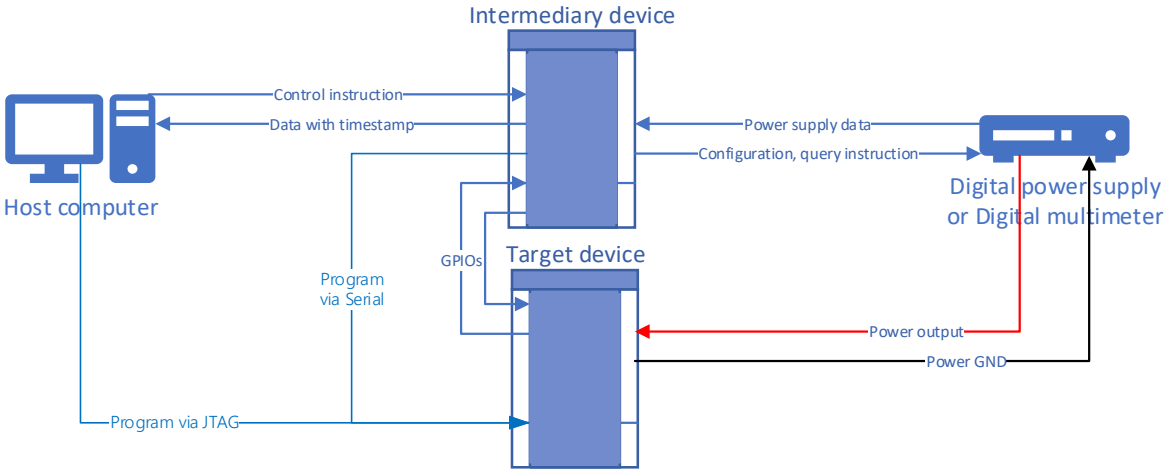


Figure 4. Connection diagram of the measuring platform.

4.2. Processor Clock Frequencies and Energy Consumption

The STM32F407 processor used as target hardware has an adjustable processor clock frequency of up to 168 MHz. In this subsection, the only variable is the processor clock frequency. Each test case is compiled with the exact same compiler optimization options and executed at different processor clock frequencies. The working current, execution time, and energy consumption under these circumstances are collected and compared. The working current of the microcontroller itself in Figure 5 shows an almost perfect linear increment with the processor clock frequency, and the approximate fitted line is

$$I(mA) = 0.2509 \times f(MHz), R^2 = 0.9961 \quad (15)$$

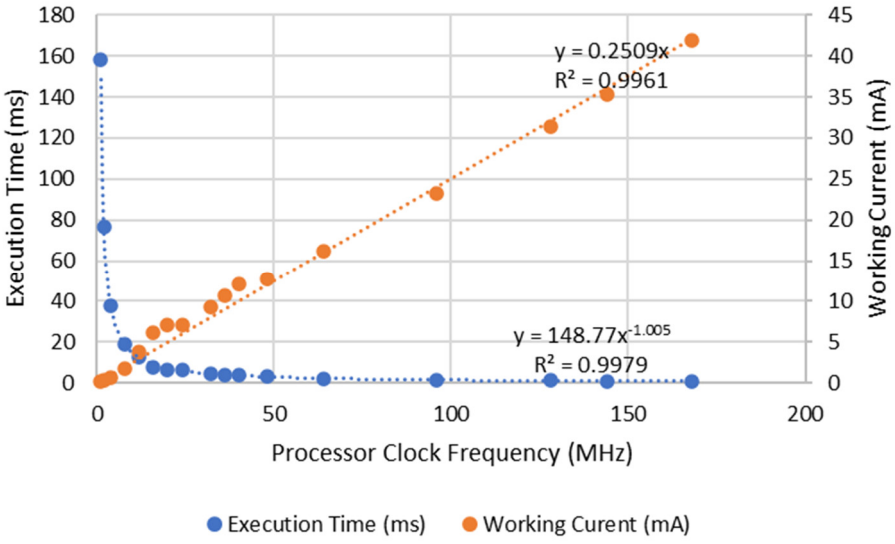


Figure 5. The relation between the execution time, the working current of the microcontroller itself, and the processor clock frequency.

According to STMicroelectronics' website, the working current of the STM32F407 processor is approximately 238 µA/MHz, and our result is consistent with the data provided by the chip vendor. Figure 5 also shows the execution time of a typical test case at different processor clock frequencies. The execution time decreases as the processor clock frequency increases, and it exhibits an inverse proportional relationship. The energy consumption of the microcontroller itself can be obtained by combining the execution time and the working current. Figure 6 shows the energy consumed by the microcontroller itself of a test case

at different processor clock frequencies. Although the working current increases with an increase in the processor clock frequency, the execution time is shortened with an increase in the processor clock frequency. As a result, the energy consumption of the microcontroller itself is hardly related to the processor clock frequency based on this test. The energy consumption of the microcontroller itself can be approximately calculated by

$$E = U \times I \times t = 5.000V \times (0.2509 \times f) \times (148.77 \times f^{-1.005}) = 186.632 \times f^{-0.005} \quad (16)$$

As mentioned in Section 3.1, the existence of peripheral components in actual applications is inevitable. Figures 7 and 8 show the execution time, the working current, and the energy consumption of the whole system.

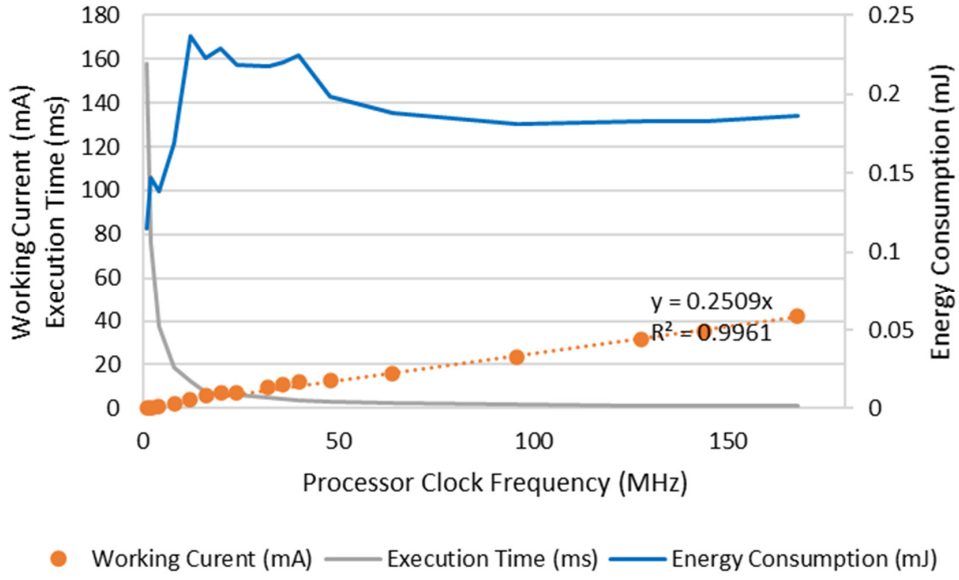


Figure 6. The relation between the energy consumption of the microcontroller itself, the processor clock frequency, and the working current.

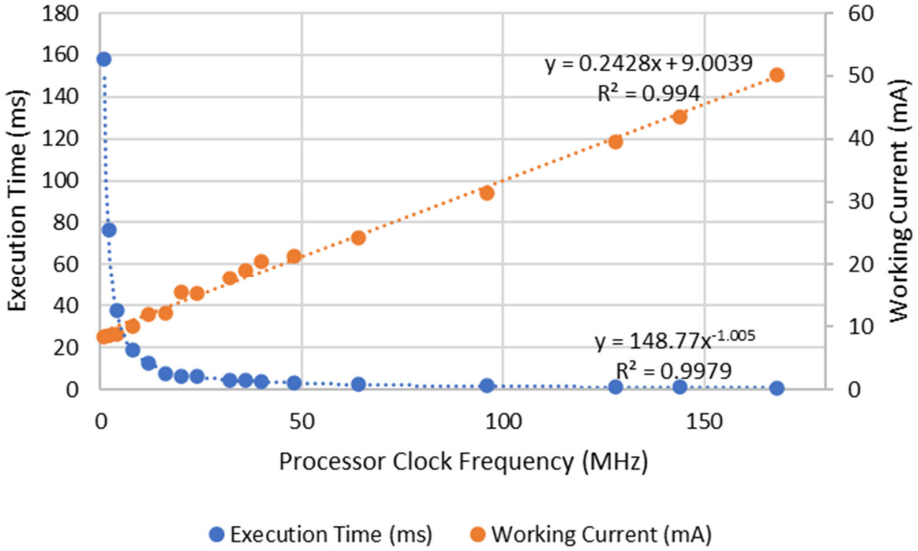


Figure 7. The relation between the execution time, the working current of the whole core board, and the processor frequency.

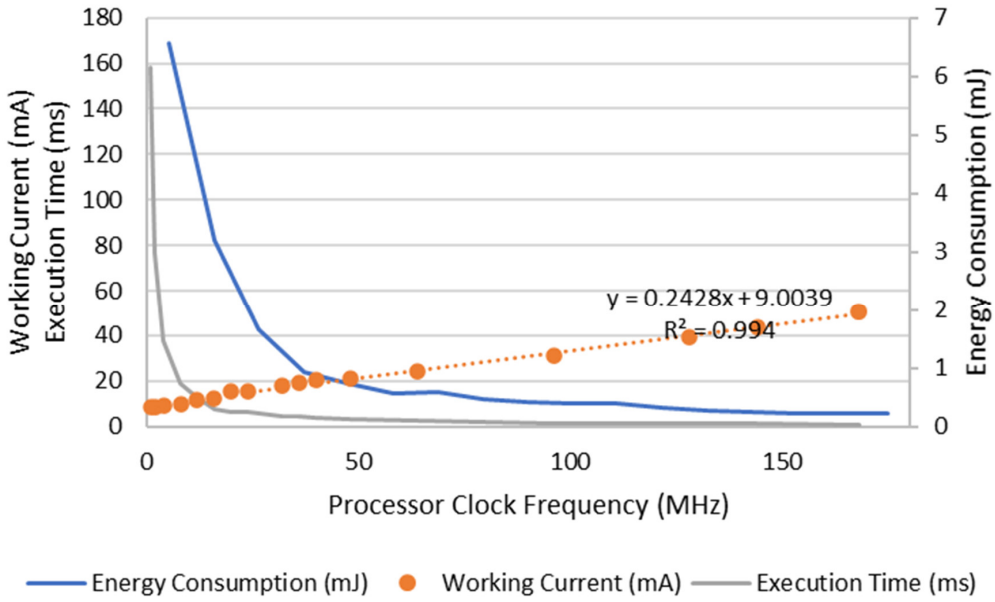


Figure 8. The relation between the energy consumption and the processor frequency of the entire core board.

It can be seen that, for the whole core board, the working current and the processor clock frequency still show a linear increment. Further, the slope, which is the working current per MHz processor clock frequency, is very close to that of the microcontroller itself. It can be concluded that the working current of the peripheral components on the core board will remain at a stable value regardless of how the processor clock frequency changes. Thus, the power consumption of peripheral components is a relatively constant value. The energy consumption of the whole system can be approximately calculated by

$$E = U \times I \times t = 5.000\ V \times (0.2428 \times f + 9.0039) \times (148.77 \times f^{-1.005}) = 180.607 \times f^{-0.005} + 6697.551 \times f^{-1.005} \quad (17)$$

It can be seen that Equation (17) is monotonically decreasing.

In this case, the energy consumption of the system consists of two parts: the energy consumed by the microcontroller itself and the energy consumed by the peripheral components. The peripheral component consumed an electric current of approximately 9 mA. By combining the results in Figures 6 and 8, we listed the energy consumed by the whole system, the microcontroller itself, and the peripheral components in Table 1. The STM32F407 processor will use the 16 MHz HSE as the processor clock if the clock configuration does not present; thus, the 16 MHz is regarded as the baseline frequency for the table data.

Table 1. Energy consumption of the whole system and peripherals at different processor clock frequencies.

HCLK (MHz)	Execution Time (ms)	Energy Consumption (mJ)			Energy Saved (%)
		Total	Processor	Peripherals	
4	37.680	1.672	0.168	1.504	−197.0
8	18.693	0.929	0.169	0.760	−65.0
16	9.350	0.563	0.223	0.340	0
32	4.660	0.414	0.217	0.197	26.5
48	3.106	0.330	0.198	0.132	41.4
64	2.326	0.282	0.188	0.094	49.9
96	1.553	0.244	0.181	0.063	56.7
128	1.165	0.230	0.183	0.047	59.1
168	0.887	0.222	0.186	0.036	60.6

According to these results, it is apparent that, in embedded systems with peripheral components, the energy consumption of computationally intensive tasks is monotonically decreasing as the processor clock frequency increases. With a higher processor clock frequency, energy is saved because the power-on duration of the peripheral components is shortened.

4.3. DVFS, Power Modes, and Their Overheads

A series of test cases are designed to investigate how much time and energy the processor will consume during processor clock frequency transitions. A rich set of test cases for DVFS technology is designed. The results are shown in Figure 9. According to our test results, some conclusions can be drawn:

(1) The time overhead of processor clock frequency transitions is relatively fixed, regardless of the difference between the previous and the latter processor frequency.
(2) The time overhead for STM32F407 to switch the processor frequency can be as high as approximately 70 ms.
(3) Considering that the peripheral clock of STM32F407 is obtained by dividing the HCLK, adjusting the processor frequency not only brings time overhead but also makes the peripherals unable to work and communicate during this period. After the processor frequency is changed, APB1 and APB2 must be reinitialized to work with the peripherals again, making the situation even worse.

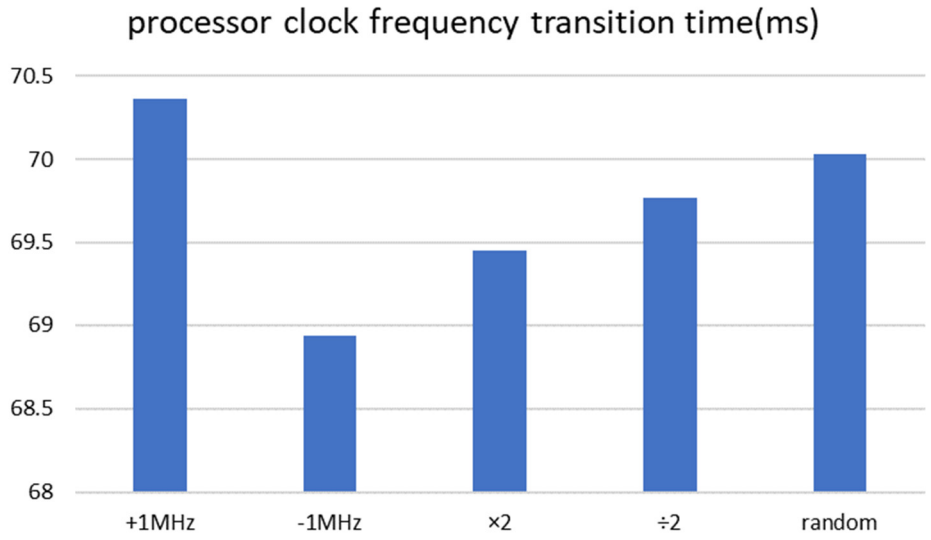


Figure 9. Time overhead of the processor clock frequency transitions for STM32F407.

The working current during the transition of the processor clock frequency is about 12.053 mA, among which about 4.075 mA is consumed by the processor, and the remaining 7.978 mA is consumed by peripheral components. Thus, the energy consumption of the processor clock frequency transition is about 4.218 mJ each time.

STM32F407 offers three different power-saving modes: sleep mode, stop mode, and standby mode. There are different regulator options in sleep mode and stop mode. Different power-saving modes with varying regulator options have different power domains turned off, which results in varying energy-saving effects. Table 2 shows the working current and power consumption of STM32F407 in different power modes with different regulator options.

Table 2. Working current of the core board in different power modes.

Mode	Regulator	Working Current (mA)	Power Consumption (mW)
Normal	N/A	12.81379	64.06895
Sleep	Main	6.25984	31.29220
	Low power	6.22557	31.12785
Stop	Main	4.82017	24.10085
	Low power	4.56274	22.81370
Standby	N/A	0.92958	4.64790

The experiment results in the previous subsection show that the power consumption of the peripheral components will not vary with different processor clock frequencies. However, from the table above, it can be seen that the power consumption of the peripheral components shows a significant change with the processor power-saving modes. The results show that standby mode has the best energy-saving performance, while sleep mode has the lowest. However, with different power domains turned off, the data loss and time cost for returning to normal mode are also different. According to the STM32F407 datasheet and user manual, standby mode can be regarded as 'power off' to some degree, which means that all data in the random access memory (RAM) and registers will be lost and can only be woken up by registered external interruptions. Stop mode will keep data in RAM and registers but will turn off the main clock generator; thus, the clock (as well as the peripheral clock) needs to be reconfigured after being woken up. We also measured the time taken to return to normal mode from sleep, stop, and standby modes with the help of external hardware, and the results are listed in Table 3.

Table 3. Time required to return from different power modes.

Power Mode	Return Time (μs)
Sleep	0.6745
Stop	5.517 (clock reconfigure excluded)
Standby	23.61 (clock reconfigure excluded)

If the DVFS technology is used to reduce power consumption when the system is idle by reducing processor clock frequency, it will require two frequency transitions as well as the overhead. The total overhead of time and energy will be about 140 ms and 8.4316 mJ. Moreover, DVFS technology will not affect the power consumption of peripheral components even though they are idle. Unlike the cost of DVFS, the cost of entering and waking up from energy-saving mode is much smaller, and it is even ignorable. Not only the power consumption of the processor but also that of peripheral components will be reduced in power-saving modes. However, at the same time, we also noticed that the processor clock would be turned off after entering stop or standby mode, and it will be reset to use HSE as the clock source after waking up. To satisfy the needs of the computing performance, the system clock requires reinitialization. Even in this case, the processor clock frequency transition is required only once, which is half of DVFS. This step will introduce an overhead of approximately 70 ms and 4.218 mJ, which is similar to that of processor clock frequency transitions.

As a low power MCU, the ESP32/Xtensa processor provides two low-power modes, namely light-sleep mode, and deep-sleep mode. In light-sleep mode, the RF module, the CPU and part of the system clock are suspended. In deep-sleep mode, all modules are closed except for RTC modules, and the wake-up procedure of deep-sleep mode is similar to the system reboot. Compared with light-sleep mode, it takes much longer to wait for the system clock to stabilize after waking up from deep-sleep mode. The working current of the ESP32 module and the wake-up time from these low-power modes to active mode are measured, and the results are listed in Table 4.

Table 4. Working current, power consumption, and wake-up time of the ESP32 module in different power modes.

Mode	Working Current (mA)	Power Consumption (mW)	Wake-Up Time (ms)
Active	50.23	251.15	N/A
Light sleep	12.21	61.05	0.44
Deep sleep	11.07	55.35	115

The waking-up period can be quite notable; thus, the working current during these periods is also measured. The results are 34.57 mA for light-sleep mode and 47.64 mA for deep-sleep mode. Consequently, the one-time energy overhead for applying light-sleep mode and deep-sleep mode is 0.078 mJ and 27.393 mJ, respectively.

5. Analysis and Suggestions

Based on our analytical models and test results, in this section, we will discuss how to achieve higher energy efficiency on embedded SoCs.

The electric current consumed while working shows a linear increase with an increase in the processor frequency, and the execution time shows an inverse proportional decrease with an increase in the processor frequency. According to the results shown in Figure 6, under ideal circumstances, when the processor is executing computationally intensive tasks, the energy consumption of the processor is not related to the clock frequency of the processor. In other words, regardless of whether the processor clock frequency increases or decreases, the processor will complete the ideal computationally intensive task with the same energy consumption, which means that E_{af} should be equal to E_{of} in Equation (2).

In the meantime, the power consumption of peripheral components remains basically constant despite different processor clock frequencies. Moreover, when the whole system enters power-saving modes, the power consumption of peripheral components is also decreased. Thus, as long as the existence of peripheral components and their power consumption is inevitable, the energy consumed by the whole system can be saved by increasing the processor clock frequency. Although increasing the processor clock frequency cannot reduce the energy consumed by the microcontroller itself in executing applications, it can shorten the power-on period of both the processor and peripheral components, thereby reducing the energy consumed by the peripheral components. In other words, for computationally intensive tasks, running at low frequency cannot save energy due to the increment of the energy consumed by the peripheral components.

Since the working current in each status is relatively constant, the energy consumption in Equation (1) can be converted into Equation (18), the sum of the energy consumption of several statuses. The energy consumption of each status is the product of the voltage, the working current, and the duration.

$$E(\mu J) = \Sigma[U(V) \times I(mA) \times t(ms)] \quad (18)$$

Take a periodic task whose task period is 100 ms as an example. The task runs on a device that is powered by a DC 5 V power supply. When the processor clock frequency is set at 16 MHz, it takes 50 ms to complete, and the system enters sleep mode in the remaining 50 ms. The energy consumption per task period is

$$E_{16MHz} = 5 \times 12.05 \times 50 + 5 \times 6.23 \times 50 = 4570\ \mu J \quad (19)$$

If the processor clock frequency is configured to 128 MHz, the task will complete in 6.25 ms, and the sleep stage will last 93.75 ms. In this case, the energy consumption per task cycle is

$$E_{128MHz} = 5 \times 39.49 \times 6.25 + 5 \times 6.23 \times 93.75 = 4154.375\ \mu J \quad (20)$$

Our results show that the transition of the processor clock frequency requires a considerable amount of time and significant energy overhead, approximately 70 ms, and 4.218 mJ each time. Thus, the DVFS technology must be carefully considered before being used. It will not only introduce additional overhead on energy consumption but also affect the real-time performance of the entire embedded system. If the energy saved by adjusting processor clock frequency is less than 4.218 mJ, or the system cannot tolerate an unresponsive period of 70 ms, utilizing DVFS technology is neither worthwhile nor suitable.

This also applies to power-saving modes. Different power-saving modes have different effects on saving energy and have different overheads on returning to normal mode. It should be noted that, when waking up from stop and standby mode, the system clock and peripheral IO clocks should be reinitialized, or it could stay at a default clock frequency. According to Table 3, the overhead of sleep mode is only 0.6745 µs, and it can be ignored, so sleep mode can be applied anywhere needed. Although stop mode and standby mode have a much better energy-saving effect than sleep mode does, if clock reconfiguration is required after waking up, sleep mode is more applicable for "short sleep". From Table 2, the power consumption of sleep mode and stop mode is about 31.1 mW and 22.8 mW. To calculate the minimum time for stop mode, Equation (13) is referenced:

$$t_{stopmin} = \frac{E_{oh}}{P_{sleep} - P_{stop}} = \frac{4.218\ \text{mJ}}{31.1\ \text{mW} - 22.8\ \text{mW}} = 508\ \text{ms} \quad (21)$$

This means only when the "sleep time" is longer than 508 ms can stop mode save more energy than sleep mode. Although standby mode has an extremely low power consumption of about 4.65 mW, all the data stored in registers and RAM will be lost. The application scenarios of standby mode are more restrictive. Even if the side effects of standby mode are ignored, compared with sleep mode,

$$t_{standbymin} = \frac{E_{oh}}{P_{sleep} - P_{standby}} = \frac{4.218\ \text{mJ}}{31.1\ \text{mW} - 4.65\ \text{mW}} = 159\ \text{ms} \quad (22)$$

The sleep time has to be longer than 159 ms to show the advantage of standby mode. Since sleep mode does not require system clock reconfiguration, it can be applied almost anywhere needed, and stop mode and standby mode can also be applied wherever HSI or HSE is used as the system clock source. However, when PLL is used as the system clock source, due to the energy consumed by clock reconfiguration, stop mode and standby mode require a minimum sleep period of about 508 ms and 159 ms, respectively. Otherwise, the energy saved will not cover the overhead of clock reconfiguration. Table 5 indicates which power-saving mode will have the best energy-saving effect under different conditions.

Table 5. Best choice of STM32F407 power-saving modes under different conditions.

Clock Source and Frequency	Memory Loss Acceptable?	Sleep Time (ms)	Best Choice
HSI/HSE	Yes	Any	Standby mode
	No	Any	Stop mode
PLL	Yes	<159	Sleep mode
		≥159	Standby mode
	No	<508	Sleep mode
		≥508	Stop mode

For the ESP32 module, the time required to wake up from light-sleep mode is significantly less than the time to wake up from deep-sleep mode. Although the light-sleep mode contributes less to power saving, the connection to the AP is maintained in this mode. Thus, if it is required to maintain the connection to the AP, the deep-sleep mode cannot be used. Our results show that waking up from deep-sleep mode requires a considerable amount of time and significant energy overhead, approximately 115 ms, and 27.393 mJ

each time. Thus, the application of deep-sleep mode should be strictly limited. Referring to Equation (13), the minimum sleep time for deep-sleep mode can be calculated as

$$t_{deep-sleepmin} = \frac{E_{oh-down} - E_{oh-light}}{P_{deep-sleep} - P_{light-sleep}} = \frac{27.393\ \text{mJ} - 0.078\ \text{mJ}}{61.05\ \text{mW} - 55.35\ \text{mW}} \approx 4.792\ \text{s} \quad (23)$$

The application of deep-sleep mode should be carefully considered, as it will not only introduce additional overhead on energy consumption but also affect the real-time performance of the entire system. This means that, if the sleep time in deep-sleep mode is shorter than 4.792 s, the energy consumption cannot be saved. Table 6 lists out which low-power mode of the ESP32 module will achieve the best energy-saving effect under different conditions.

Table 6. Best choice of ESP32 low-power modes under different conditions.

Sleep Time (s)	Best Choice
<4.792	Light sleep
≥4.792	Deep sleep

6. Future Directions

Two widely used embedded microcontrollers of different types and architectures were selected for this study. According to the experiment results, our analytical models work well with both processors. However, we do not have the conditions to significantly change the ambient temperature, and the temperature of the STM32F407 processor never exceeded 35 °C by its self-heating. Thus, the effects of environmental temperature on power and energy consumption have not been fully measured and analyzed. Due to the diversity of operating environments, we believe that it is worthwhile to select a wider range of microcontrollers of different types and architectures, as well as in various external environments, to take a deeper look.

7. Discussion and Conclusions

In this paper, we review many power-saving technologies and propose our assumptions on their impacts on energy consumption. To validate our assumptions, we designed two analytical models and a rich set of test cases. The experiment results proved our assumptions are correct, and we drew some conclusions about the application scenarios of power-saving technologies.

According to our experiment results, we can answer the questions that we raised: first, in the case of computationally intensive tasks, the energy consumption of the processor is not related to processor clock frequency; however, due to the existence of on-board peripheral components, increasing processor clock frequency will help to reduce the total energy consumption of the whole system, since increasing frequency will shorten the power-on time of the system. Second, the time and energy overheads of clock reconfiguration introduced by applying DVFS or waking up from some power-saving modes are fairly significant. For example, the STM32F407 processor used in this paper requires a "long sleep" to cover the overheads, which are approximately 70 ms and 4.218 mJ. We have also drawn equations for calculating the minimum sleep time, through which we can find the most appropriate and energy-efficient power-saving technology in a specified situation.

Hardware designers should try their best to allow all peripheral components to be powered off separately. Software developers should consider having these peripheral components powered off or deactivated once the peripheral components have finished their jobs or are temporarily not required. The results also present some interesting information. Unlike conventional ideas, increasing the processor clock frequency results in higher power consumption, but it does not necessarily mean more energy consumption or poorer energy efficiency. In this way, if two processors have similar working currents per MHz processor

clock frequency, choosing the one with a higher designed frequency may help achieve better energy efficiency.

Author Contributions: Methodology, H.W.; software, H.W.; validation, H.W.; investigation, H.W.; writing—original draft preparation, H.W.; writing—review and editing, K.W.; supervision, C.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dar, K.S.; Asif, S.; Islam, A. Power Management and Green Computing: An Operating System Prospective. *Can. Int. J. Soc. Sci. Educ.* **2015**, *2*, 164–183.
2. Thakkar, A.; Chaudhari, K.; Shah, M. A Comprehensive Survey on Energy-Efficient Power Management Techniques. *Procedia Comput. Sci.* **2020**, *167*, 1189–1199. [CrossRef]
3. Adegbija, T.; Rogacs, A.; Patel, C.; Gordon-Ross, A. Microprocessor Optimizations for the Internet of Things: A Survey. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2018**, *37*, 7–20. [CrossRef]
4. Sun, L.; Sheshadri, R.K.; Zheng, W.; Koutsonikolas, D. Modeling WiFi Active Power/Energy Consumption in Smartphones. In Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems, Madrid, Spain, 30 June–3 July 2014; pp. 41–51.
5. Martinez, B.; Montón, M.; Vilajosana, I.; Prades, J.D. The Power of Models: Modeling Power Consumption for IoT Devices. *IEEE Sens. J.* **2015**, *15*, 5777–5789. [CrossRef]
6. Yoon, C.; Lee, S.; Choi, Y.; Ha, R.; Cha, H. Accurate power modeling of modern mobile application processors. *J. Syst. Archit.* **2017**, *81*, 17–31. [CrossRef]
7. Alawnah, S.; Sagahyroon, A. Modeling of smartphones' power using neural networks. *EURASIP J. Embed. Syst.* **2017**, *2017*, 22. [CrossRef]
8. Anantha, C.P.; Denis, D.C.; Joyce, D.C.; Yogesh, R.K. Next generation micro-power systems. In Proceedings of the 2008 IEEE Symposium on VLSI Circuits, Honolulu, HI, USA, 18–20 June 2008; pp. 2–5.
9. Gollakota, S.; Reynolds, M.S.; Smith, J.R.; Wetherall, D.J. The Emergence of RF-Powered Computing. *Computer* **2014**, *47*, 32–39. [CrossRef]
10. Gudan, K.; Chemishkian, S.; Hull, J.J.; Reynolds, S.M.; Thomas, S. Feasibility of wireless sensors using ambient 2.4GHz RF energy. In Proceedings of the 2012 IEEE SENSORS, Taipei, Taiwan, 28–31 October 2012.
11. Li, C.; Zhang, W.; Cho, C.B.; Li, T. SolarCore: Solar energy driven multi-core architecture power management. In Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture, San Antonio, TX, USA, 12–16 February 2011; pp. 205–216.
12. Raghunathan, V.; Kansal, A.; Hsu, J.; Friedman, J.; Srivastava, M. Design considerations for solar energy harvesting wireless embedded systems. In Proceedings of the IPSN 2005 Fourth International Symposium on Information Processing in Sensor Networks, Boise, ID, USA, 15 April 2005; p. 64.
13. Hua, S.; Qu, G. Approaching the maximum energy saving on embedded systems with multiple voltages. In Proceedings of the IEEE/ACM International Conference on Computeraided Design, San Jose, CA, USA, 9–13 November 2003.
14. Gheorghita, S.V.; Basten, T.; Corporaal, H. Application Scenarios in Streaming-Oriented Embedded System Design. In Proceedings of the International Symposium on System-on-Chip, Tampere, Finland, 14–15 November 2006.
15. Choi, K.; Soma, R.; Pedram, M. Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **2004**, *24*, 18–28. [CrossRef]
16. Saewong, S.; Rajkumar, R. Practical voltage-scaling for fixed-priority RT-systems. In Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium, Toronto, ON, Canada, 30 May 2003.
17. Quan, G.; Hu, X. Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors. In Proceedings of the 38th Annual Design Automation Conference, Taipei, Taiwan, 9–14 October 2001.
18. Kan, E.Y.Y.; Chan, W.K.; Tse, T.H. Leveraging Performance and Power Savings for Embedded Systems Using Multiple Target Deadlines. In Proceedings of the 2010 10th International Conference on Quality Software, Zhangjiajie, China, 14–15 July 2010.
19. STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 Advanced Arm®-based 32-bit MCUs—Reference Manual. Available online: https://www.st.com/resource/en/reference_manual/dm00031020-stm32f405415-stm32f407417-stm32f427437-and-stm32f429439-advanced-armbased-32bit-mcus-stmicroelectronics.pdf (accessed on 8 March 2019).

20. Li, D.; Chou, P.H.; Bagherzadeh, N. Mode selection and mode-dependency modeling for power-aware embedded systems. In Proceedings of the ASP-DAC/VLSI Design 2002, 7th Asia and South Pacific Design Automation Conference and 15h International Conference on VLSI Design, Bangalore, India, 11 January 2002.
21. Hoeller, A.S.; Wanner, L.F.; Fröhlich, A.A. A Hierarchical Approach for Power Management on Mobile Embedded Systems. In Proceedings of the IFIP Working Conference on Distributed and Parallel Embedded Systems, Braga, Portugal, 11–13 October 2006.
22. Huang, K.; Santinelli, L.; Chen, J.; Thiele, L.; Buttazzo, G.C. Adaptive power management for real-time event streams. In Proceedings of the 2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC), Taipei, Taiwan, 18–21 January 2010.
23. Bhatti, K.; Belleudy, C.; Auguin, M. Power Management in Real Time Embedded Systems through Online and Adaptive Interplay of DPM and DVFS Policies. In Proceedings of the 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, Hong Kong, China, 11–13 December 2010.
24. Niu, L.; Quan, G. Reducing both dynamic and leakage energy consumption for hard real-time systems. In Proceedings of the 2004 international conference on Compilers, Architecture, and Synthesis for Embedded Systems, Washington, DC, USA, 22–25 September 2004.
25. Liang, J.M.; Chen, J.J.; Cheng, H.H.; Tseng, Y.C. An energy-efficient sleep scheduling with QoS consideration in 3GPP LTE-advanced networks for Internet of things. *IEEE J. Emerging Sel. Topics Circuits Syst.* **2013**, *3*, 13–22. [CrossRef]
26. Ahmad, R.W.; Gani, A.; Ab Hamid, S.H.; Naveed, A.; Ko, K.; Rodrigues, J.J. A case and framework for code analysis–based smartphone application energy estimation. *Int. J. Commun. Syst.* **2016**, *30*, e3235. [CrossRef]
27. Chen, N. Research and Implementation of Low Power Technology for Embedded Real-Time Operating System. Master Thesis, University of Electronic Science and Technology of China, Chengdu, China, 2015.
28. Optimize Options (Using the GNU Compiler Collection (GCC)). Available online: https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html (accessed on 31 July 2018).
29. Arm Compiler User Guide. Available online: https://developer.arm.com/documentation/100748/0610 (accessed on 31 July 2018).
30. Chung, E.Y.; Benini, L.; Micheli, G.D. Source code transformation based on software cost analysis. In Proceedings of the 14th International Symposium on Systems Synthesis, Montreal, QC, Canada, 30 September–3 October 2001.
31. Dzhagaryan, A.; Milenković, A. Impact of thread and frequency scaling on performance and energy in modern multicores: A measurement-based study. In Proceedings of the 2014 ACM Southeast Regional Conference, Kennesaw, GA, USA, 28–29 March 2014.
32. Kim, W.; Gupta, M.S.; Wei, G.; Brooks, D. System level analysis of fast, per-core DVFS using on-chip switching regulators. In Proceedings of the 2008 IEEE 14th International Symposium on High Performance Computer Architecture, Salt Lake City, UT, USA, 16–20 February 2008.
33. Jiangwei, H. Partitioning the Program into different regions using dynamic and Static Approach with kernel-Assisted in Power management for Embedded System. In Proceedings of the 2006 IEEE International Conference on Information Reuse & Integration, Waikoloa, HI, USA, 16–18 September 2006.
34. Pinheiro, D.; Gonc¸alves, R.; Valentin, E.; Oliveira, H.; Barreto, R. Inserting DVFS Code in Hard Real-Time System Tasks. In Proceedings of the 2017 VII Brazilian Symposium on Computing Systems Engineering, Curitiba, Brazil, 6–10 November 2017; pp. 23–30.
35. Kuehn, P.J.; Mashaly, M. DVFS-Power Management and Performance Engineering of Data Center Server Clusters. In Proceedings of the 2019 15th Annual Conference on Wireless On-demand Network Systems and Services (WONS), Wengen, Switzerland, 22–24 January 2019; pp. 91–98.
36. Wu, H. An Automatic Energy Consumption Measuring Platform for Embedded Systems. In Proceedings of the 2019 6th International Conference on Information Science and Control Engineering (ICISCE), Shanghai, China, 20–22 December 2019.
37. Wu, H. Two Designs of Automatic Embedded System Energy Consumption Measuring Platforms Using GPIO. *Appl. Sci.* **2020**, *10*, 4866. [CrossRef]
38. Series 2280S Precision Measurement, Low Noise, Programmable DC Power Supplies Datasheet. Available online: https://download.tek.com/datasheet/1KW-50894-0_2280S_Datasheet_040219.pdf (accessed on 10 June 2017).
39. STM32F407/417-STMicroelectronics. Available online: https://www.st.com/en/microcontrollers-microprocessors/stm32f407-417.html (accessed on 8 March 2019).