Experimental Teaching Reform to Embedded System Curriculum

Xing Liu
Hubei Key Laboratory of Transport IoT
Wuhan University of Technology
Wuhan, China
liu.xing@whut.edu.cn

Yaobang Zhao
Computer Science and Technology
Wuhan University of Technology
Wuhan, China
jasonchio@whut.edu.cn

Jingling Yuan
Computer Science and Technology
Wuhan University of Technology
Wuhan, China
yjl@whut.edu.cn

Wenbi Rao
Computer Science and Technology
Wuhan University of Technology
Wuhan, China
wbrao@whut.edu.cn

Liping Lu *
Computer Science and Technology
Wuhan University of Technology
Wuhan, China
luliping@whut.edu.cn

Fei Huang
Western Research Institute
Yangtze University
Xinjiang, China
whutcst@yeah.net

***Abstract*—With the rapid development of embedded technology, the traditional experiments of embedded system curriculum meet several challenges, such as the experiment design is lack of comprehensive perspective, the experimental program is insufficient to cultivate the students' independent development ability, and the teaching contents cannot keep up with the development trend of embedded technology. To address these challenges, an experimental teaching reform to the embedded system curriculum is implemented in this paper. To enable the students to build a comprehensive perspective to the embedded system architecture, five courses related to the embedded system are integrated to construct a complete embedded architecture ranging from the underlying hardware to the upper application. To strengthen the students' independent development ability, a real-time multithreaded embedded OS is required to be developed by the students. To keep the teaching contents up with the development trend of embedded technology, some emerging technologies such as multi-core embedded architecture, LoRa, NB-IoT, etc., are proposed and used as the experimental cases. The experimental teaching reform targets to enable the students to establish a comprehensive embedded system perspective, become proficient in the embedded software and hardware skills, cultivate the cross-layer software and hardware collaborative optimization thinking, and master the way to solve some difficulties in the embedded system. The teaching reform has been implemented in the Internet-of-Things speciality for four years, and the survey results proved it had achieved the expected teaching effects effectively.**

***Index Terms*—embedded system, embedded operating system, experiment, teaching reform**

I. Introduction

Embedded system is a kind of application-centric system which can flexibly tailor the software and hardware modules in terms of the application requirements. Embedded system technology has been widely applied in daily life, e.g., the commonly used smart phones, smart watches and wearable devices have used the embedded technology. Due to the widespread applications, embedded system curriculum becomes essential in the computer science and electronic engineering specialities [1]. Recently, as the embedded technology develops rapidly, how to teach the embedded system curriculum well to enable the students to keep competitive in the future becomes a topic worth investigating.

Experimental teaching is significant to embedded system curriculum as it can deepen the students' understanding of the theoretical knowledge. Commonly, the experiment program of embedded system is to teach the students to master the operating skills such as GPIO programming, interruption, serial communications, A/D conversion, timer, watchdog, boot loader, operating system (OS) porting, driver development and so on [2]–[4]. This experimental teaching method is effective to make the students master the basic development skills of embedded system, yet it has the following limitations:

First, the experiment design is lack of comprehensive perspective. A comprehensive embedded system architecture should range from the lowest level of embedded hardware, to the upper level of embedded operating system, to the higher level of network protocol, database and middleware, and finally to the highest level of embedded applications. Although the current embedded system curriculum has opened many courses corresponding to the above contents, such as the courses of embedded system development, embedded operating system and communication technology, the teaching contents of these courses are commonly not closely interconnected with each other. Different courses use different experimental platforms and different experimental cases, making it not conducive to cultivate the students' cognition of comprehensive embedded architecture, not to mention cultivating the students' cross-layer collaborative optimization thinking way.

Second, the experimental program does not pay enough attention to train the students' independent development ability. Many experimental cases focus on the porting, configuration and verification of some existing work, but not require the students to design and implement an embedded system inde-

* Author to whom all the correspondences should be addressed.

978-1-7281-7267-5/20/$31.00 ©2020 IEEE

pendently by their own knowledge. Consequently, the students do not have sufficient practice of the theoretical knowledge, and are even not aware of the problems that will occur during the development process. This teaching method is not conducive to enhancing the students' competitiveness in the embedded system field.

Regarding to the above limitations, a teaching reform to the embedded system experiments is implemented by adopting the following concepts: (1) To enable students to build a comprehensive perspective of the embedded systems, five courses related to embedded systems are integrated. These courses are the microcontroller principle, embedded system development, embedded OS, wireless sensor network (WSN) and embedded application design. These five courses correspond to the four different levels of embedded system architecture, which are the hardware, OS, network protocol stack and applications. The experiments of each course are designed on the basis of previous course's experimental results, and all the experiments of the five courses are interlinked to ultimately constitute a complete embedded architecture. (2) To cultivate the students' independent development ability, the students no more just port an existing embedded OS to the embedded device and verify its functionalities. Instead, they are required to implement a new real-time multithreaded embedded OS independently by their own knowledge.

The teaching reform in this paper targets to enable the students to establish a comprehensive viewpoint to the embedded system, become proficient in the embedded software and hardware skills, cultivate the cross-layer software and hardware collaborative optimization thinking, and become competent to solve some difficult problems in the embedded systems.

The rest of this paper is organized as follows: Section II introduces the related work of the embedded system teaching. Section III presents the overall experimental design for the five courses related to the embedded system. It discusses how to integrate the experiments of these five courses so as to build a comprehensive embedded architecture. Section IV investigates the experimental cases and experimental objective of each course. Section V does a survey on the teaching effects of this experimental reform. Finally in section VI, the conclusion is given.

II. Related Work

Many embedded system experiments target to enable the students to master the basic software and hardware operating skills. Rankovska et al. [2] taught the embedded system course by developing a flexible educational model which could help students learn the most common methods for the web-based control and monitoring. He et al. [3] used the small but powerful BeagleBone Black (BBB) board to enrich the teaching materials of embedded system courses, and proved the BBB was an easy-to-use development board for developing complex embedded systems. Yakimov et al. [4] used the open-source Arduino development board and IDE software to convince the students in the flexibility and universality of embedded systems.

Some researchers teach the embedded system experiments by building the known embedded microcontrollers on the reconfigurable FPGA, which is novel and interesting. Stanciu et al. [5] presented the possibility of emulating different known microcontrollers on the same FPGA development board to enable the students learn different microprocessor-based embedded systems in a comparative approach.

Some experimental work targets to train the students' hardware and software collaborative design thinking. Bencheva et al. [6] presented a case study which involved the design of a simple but realistic embedded system using hardware/software co-design approach, and the results showed this course had enhanced the students' understanding of practical problems related to hardware/software co-design.

Many other teaching reforms are implemented to the embedded system course to improve the teaching quality and prompt the students' study interests. Wu et al. [7] proposed the autonomous learning mode for the embedded system course, which made the students become the main body of teaching activity while the teachers provided guidance by key concept explain, problem analysis, and other methods. Hong et al. [8] implemented the teaching reform by emphasizing *learning* rather than *teaching*, *practice* rather than *verification*, and the results showed this way improved the students' practical innovation ability. Wang et al. [9] implemented a series of micro videos for real-time embedded OS course to facilitate students' understanding and practice, and the results showed the students gave quite positive comments on the video assistants.

III. Overall Experimental Design

This section presents the overall experimental design for the five integrated embedded courses. This integration of the five courses target to enable the students to build a comprehensive embedded architecture and cultivate the students' cross-layer software and hardware collaborative optimization thinking. The overall design is depicted in Figure 1.

First, the microcontroller principle course is opened. Through the experiments of this course, the students can master some basic hardware development skills such as GPIO programming, interruption, serial communication and so on. At the end of this course, the students are required to develop two projects on the STM32 board. One is the voice recording and playback project, and the other is the electronic organ project.

Next, based on the hardware knowledge and operating skills learned from the last course, the students learn how to run an OS on the microcontroller, and this objective is accomplished by the embedded system development course. This course targets to teach the students to run the embedded Linux system on the microcontroller, and the experimental topics include the setup of cross-compile toolchain, the development of universal boot loader, the porting of embedded Linux kernel, the usage of busybox file system, the development of Linux drivers and

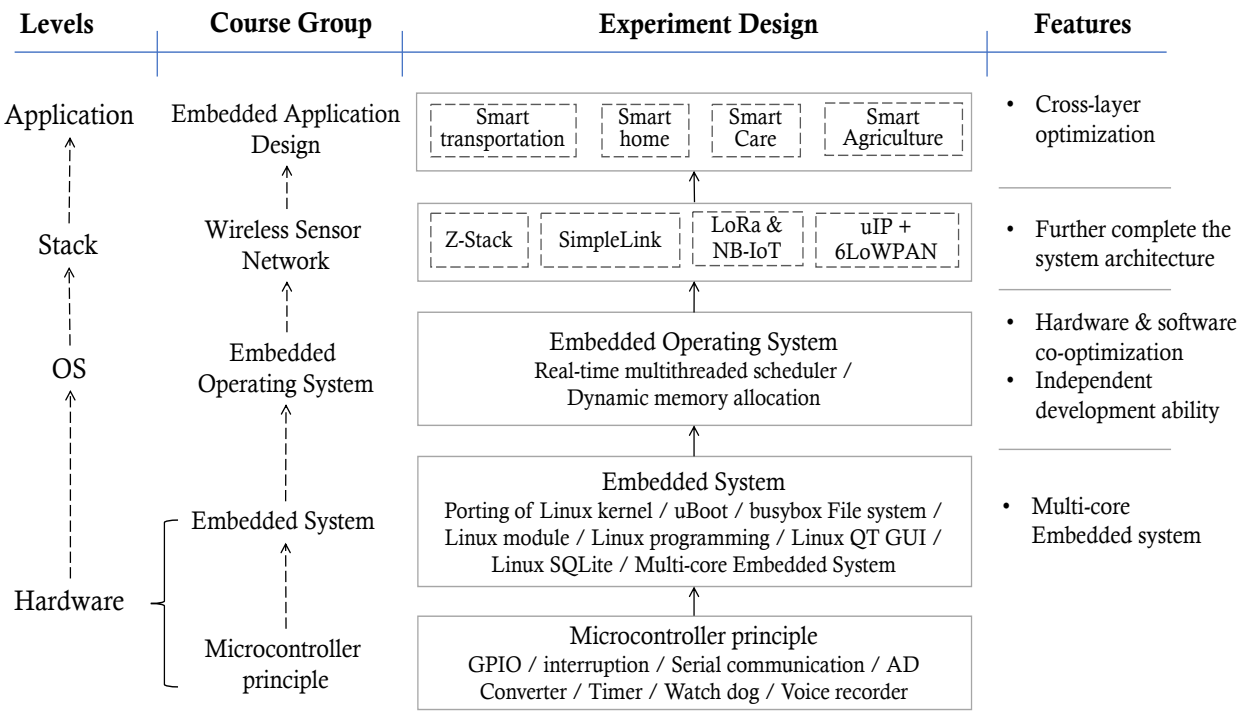

Fig. 1. Overall experimental design for the five embedded related courses.

modules, Linux programming, Linux QT GUI development, and the ARM big.LITTLE multi-core embedded system.

Through the above two courses, the students master the way to run the embedded Linux on the microcontroller. Yet, just porting the Linux OS is insufficient to enhance the students' embedded development ability. The students should be able to design and implement an embedded OS independently by their own knowledge. Therefore, the embedded OS course is opened further, and its objective is to teach the students to develop a new real-time multithreaded embedded OS dedicated to the resource-constrained embedded devices. Through the experiments of this course, the students' independent development ability can be enhanced.

Through the above three courses, the students can run the OS on the embedded systems. Based on the OS, the embedded system can run many stand-alone applications such as peripheral control and environment monitoring. Yet, they can still not interact with the other devices. Therefore, in the next step, the students should port the network protocols to the OS that they have built so as to enable the embedded devices to communicate with each other, and this objective is accomplished by the wireless sensor network (WSN) course. Through the WSN experiments, the students learn the features of different network protocols such as Z-stack, LoRa, NB-IoT and uIP, and master the way to port them to the embedded OS.

With the above four courses, an embedded system architecture which includes the hardware, OS and network protocol has been constructed. Then, the embedded application design course is opened. This course requires the students to integrate the knowledge of the previous four courses, and use the cross-layer software and hardware collaborative optimization approaches to solve some difficulties in the embedded applications.

TABLE I
EXPERIMENT DESIGN FOR MICROCONTROLLER PRINCIPLE COURSE.

No	Topics	Experiment Cases
1	GPIO programming	Marquee LED blink
2	Interruption	Keyboard scan
3	Serial Communication	Dual serial communication
4	Timer and watchdog	Digital clock
5	A/D converter	Signal generator
6	Comprehensive experiment	Electronic organ Voice recording and playback

IV. EXPERIMENTAL DESIGN FOR EACH COURSE

In this section, the experimental proposals of the five embedded related courses are presented respectively.

A. Experiment Design for Microcontroller Principle

The experimental objective of microcontroller principle course is to enable the students to master the basic hardware knowledge and operating skills of microcontrollers.

The experimental topics of this course are depicted in Table I. It includes the setup of the development environment, the GPIO programming, the interruption handling, the serial communications, the timer and watchdog and the A/D converter. In addition, two comprehensive projects are proposed and require the students to complete, one is the voice recorder and playback project and the other is the electronic organ project.

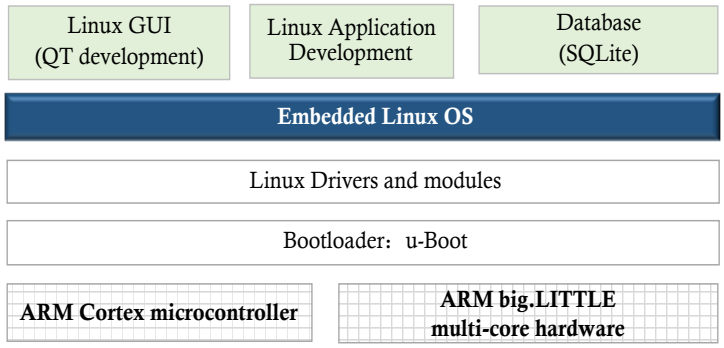

Fig. 2. Experiment design for embedded system development course.

B. Experiment Design for Embedded System Development

The embedded system development course is based on the microcontroller principle course, and its experimental objective is to enable the students to run the embedded Linux on the microcontroller.

In Figure 2, the experimental programs of the embedded system development course are depicted. There are six experimental cases, and they are the usage of universal boot loader, the porting of Linux kernel to ARM Cortex microcontroller, the development of Linux drivers and modules, the Linux programming, the Linux GUI development, and the porting of embedded database SQLite to Linux, respectively.

In addition, the ARM big.LITTLE multi-core experiment is also proposed [10], and the purpose of this experiment is to enable the students to understand the basic multi-core embedded knowledge and learn how to use multi-core technology to optimize the energy consumption of the embedded devices. The learning of the multi-core technology in this course will lay the foundation for the experiment teaching of the next embedded operating system course.

C. Experiment Design for Embedded Operating System

The objective of embedded OS experiments is to enable the student to strengthen the independent development ability by developing a real-time multithreaded embedded OS dedicated to the resource-constrained embedded devices. Three experimental cases are designed for this courses, and they are respectively the real-time multithreaded scheduler, the dynamic memory allocator and the timer.

a) Experiment case 1 - Real-time multithreaded scheduler: This experiment requires the students to develop a real-time multithreaded scheduler by implementing the RMS (Rate Monotonic Scheduling) algorithm. Multithreaded scheduler can support the task preemption through which the high-priority task can be executed in real time by preempting the low-priority ones. Yet, when a task is preempted, its run-time contexts need to be saved so as to resume its execution later. Therefore, each thread needs an independent run-time stack. In Figure 3, the structure of the thread control block (TCB) is defined. The stack pointer in TCB points to the address of the stack top, and the function pointer in TCB points to the handler to be executed when the thread starts execution. Different scheduling algorithms can be applied for dispatching the threads, such as the Round-Robin (RR) algorithm, the RMS algorithm, and the EDF (Earliest Deadline First) algorithm. In this experiment, the students are required to implement the RMS algorithm for the thread scheduler. Thus, the TCBs of all the threads can be linked in the order of the thread priority. When a thread posts a request to dispatch the scheduler, the scheduler will check the thread status from the head of the TCB link list to the tail of the TCB link list until a thread is found to be the status of active. Then, this thread can resume its execution.

b) Experiment case 2 - Dynamic memory allocator: Dynamic memory allocation can improve the memory utilization efficiency and it is critical for the embedded OS since most embedded devices are constrained in the memory resources. In this experiment case, two kinds of dynamic memory allocators are required to be implemented by the students, one is the fixed-size block allocation, and the other is the non-fixed size allocation. In Figure 4, the dynamic memory allocator with fixed-size blocks is depicted. The memory heap is divided into a set of partitions, and each partition is divided further into a set of fixed-size blocks. The block size in each partition is different and can be configured in terms of the application contexts. Each partition has a free list which links all the free blocks. When performing the allocation, the allocator deletes one block from the head of the free list. And when performing the deallocation, the allocator inserts the deallocated block to the tail of the free list. The advantage of this allocation is the allocation operation can be completed with a fixed time, and this is critical for the real-time system in which the operation time needs to be predictable. Yet, the memory utilization efficiency of this allocation is not high as the allocated size is fixed and it can be larger than the actual required size. In Figure 5, the non-fixed size memory allocation mechanism is depicted. The same as the fixed-size allocation, a free list is used to manage the free memory blocks in the heap. Yet, due to the non-fixed size allocation, when performing the allocation, it needs to search for an appropriate block from the free list. And when performing the deallocation, the adjacent free blocks need to be merged. As it is shown in Figure 5.a, after block A is deallocated and block E is allocated, the free list will be updated as is shown in Figure 5.b. The advantage of

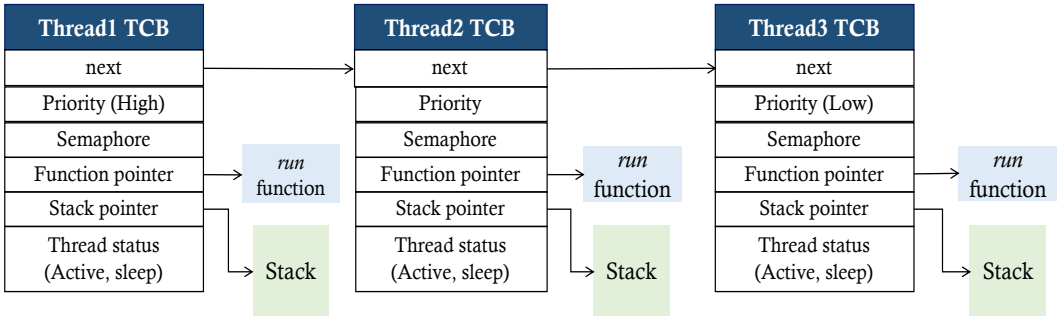

Fig. 3. Thread control block for the real-time multithreaded scheduler.

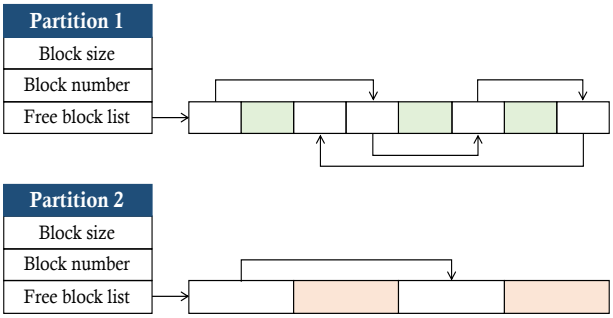

Fig. 4. Dynamic memory allocator with fixed-size blocks.

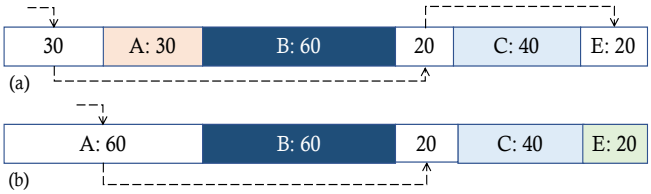

Fig. 5. Dynamic memory allocator with non-fixed size management.

the non-fixed size allocation is that the memory resources can be utilized more efficiently than the fixed-size case. However, the allocation time is not fixed and it depends on the searching time on the free list. Thus, this kind of allocation is not appropriate for some highly time-constrained real-time OS.

c) Experiment case 3 - Timer: The students are also required to develop a timer for the embedded OS. The timer list is depicted in Figure 6. Each timer has an initial timer interval which indicates the intervals to trigger this timer. A time interruption is set in the system. Each time when the interruption is activated, the intervals of each timer will be updated by subtracting the interrupt trigger time. When the intervals decrease to zero, the timer is triggered. Then, the callback function will be executed. Two executing modes can be set for the timer, one is one slot timer mode, and the other is the periodic mode. For the latter one, the interval value will be restored to the initial one when it decreases to zero. All timers are linked by a timer list. A new timer can be added to this list by the function $start\_timer$, and an existed timer in this list can be deleted by the function $stop\_timer$.

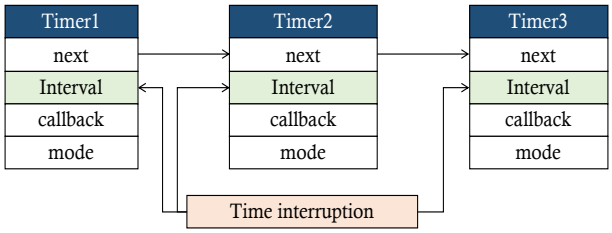

Fig. 6. Timer management list.

D. Experiment Design for Wireless Sensor Network

In Table II, the experimental programs of WSN course are depicted. Four experimental topics are proposed, and they are the Z-stack, LoRa, NB-IoT and uIP. The students need to understand the different features, advantages and drawbacks of these protocols, and then select one appropriate protocol in terms of the application contexts. Later, they need to port the selected protocol to the embedded OS that they have developed in the previous embedded OS course.

TABLE II
EXPERIMENT DESIGN FOR WSN COURSE.

No.	Topics	Features
1	Z-stack	ZigBee standard (IEEE 802.15.4)
2	LoRa	low power, Long Range Radio, LPWAN
3	NB-IoT	Narrow Band Internet of Things, LPWAN
4	uIP + 6LowPAN	TCP/IP standard

E. Experiment Design for Embedded System Application

In the embedded application design course, four kinds of application cases are proposed, and they are the smart transportation, smart home, smart care and smart agriculture. The students can select one application based on personal interests, and then use the knowledge and skills that they have learned from the previous four courses to accomplish the application tasks. For each application, the following performance requirements should be satisfied: (1) Real-time response. The embedded devices should reactive quickly to the events. (2) Low memory requirement. The software system should have low memory cost. (3) Low power. Devices should have long lifetime even without energy charging.

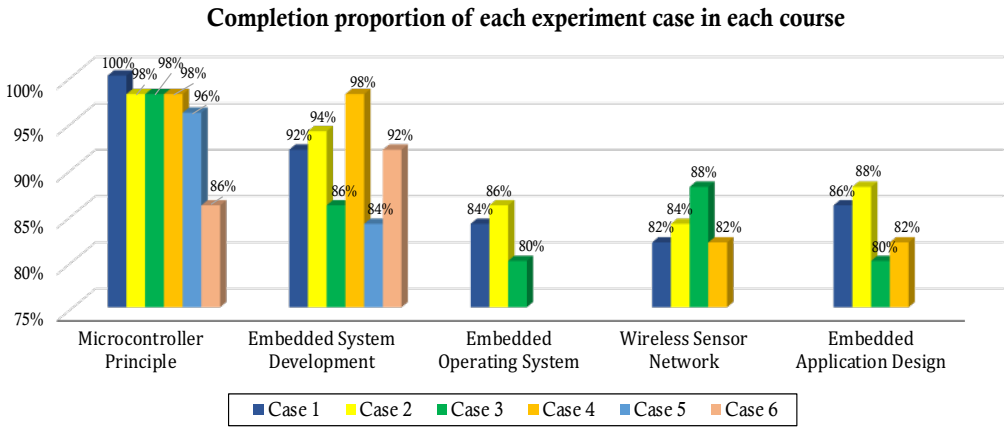

Fig. 7. Completion proportion of each experiment case in each course.

This course is comprehensive and can cultivate the students to use the cross-layer software and hardware collaborative design approaches to solve some embedded difficulties.

V. Course Survey

To evaluate the teaching effects of the experimental reform, we conduct a survey on the completion rate of each experimental case in each course, and the results are shown in Figure 7. It is shown from the results that more than 80% students can develop a multithreaded embedded OS under guidance, more than 82% students can integrate the network protocols with their embedded OS, and more than 80% students can complete the embedded application tasks by integrating the knowledge of different courses. The survey results confirmed the effectiveness of the experimental reform.

VI. Conclusions

This paper presents the teaching reform to the embedded system curriculum which integrates the experimental teaching of five embedded related courses: microcontroller principle, embedded system development, embedded OS, WSN and embedded application design. This teaching reform targets to enable the students to establish a comprehensive perspective to the embedded system, become proficient in the embedded software and hardware skills, cultivate the cross-layer software and hardware collaborative optimization thinking, as well as be competent to solve some embedded difficulties. This teaching reform has been implemented in the Internet-of-Things speciality for four years, and the survey results proved it has achieved its initial teaching objective effectively.

Acknowledgment

Our thanks to the support from Natural Science Foundation of Xinjiang Province (Grant No. 2020D01A130), the support from National Natural Science Foundation of China (No. 61702387), and the support from the 2020 Teaching Research Project of Wuhan University of Technology entitled ”Teaching Reform to the Experiments of Fundamental Computer Software and Hardware Course Groups Oriented to Complex Engineering Problems”.

References

[1] IBRAHIM, Intisar, et al. Embedded systems pedagogical issue: Teaching approaches, students readiness, and design challenges. American Journal of Embedded Systems and Applications, 2015, 3.1: 1-10.

[2] RANKOVSKA, Valentina Vasileva. Web-Based Monitoring and Control in Embedded Systems Teaching. In: 2019 IEEE XXVIII International Scientific Conference Electronics (ET). IEEE, 2019. p. 1-4.

[3] HE, Nannan; QIAN, Ying; HUANG, Han-way. Experience of teaching embedded systems design with BeagleBone Black board. In: 2016 IEEE International Conference on Electro Information Technology (EIT). IEEE, 2016. p. 0217-0220.

[4] YAKIMOV, Peter Ivanov. Open-source platforms application in introductory embedded systems teaching. In: 2016 XXV International Scientific Conference Electronics (ET). IEEE, 2016. p. 1-4.

[5] STANCIU, Alexandra, et al. Reconfigurable platform for embedded systems teaching. In: 2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME). IEEE, 2017. p. 455-458.

[6] BENCHEVA, Nina; KOSTADINOV, Nikolay. Teaching Hardware/Software Co-design of Embedded Systems–a Case Study. In: 2017 27th EAEEIE Annual Conference (EAEEIE). IEEE, 2017. p. 1-2.

[7] WU, Rui; LIU, Hongwei; YIN, Fang. Teaching model reform and practice in embedded system experiment. In: 2015 10th International Conference on Computer Science & Education (ICCSE). IEEE, 2015. p. 889-892.

[8] HONG, Guo, et al. Teaching Reform of Embedded System Course Based on Innovation Ability Cultivation. In: 2018 9th International Conference on Information Technology in Medicine and Education (ITME). IEEE, 2018. p. 632-635.

[9] WANG, Zhaohong; MEEHAN, Kathleen; GUO, Jing. Teaching with Video Assistance in Embedded Real-Time Operating System. In: 2018 IEEE Frontiers in Education Conference (FIE). IEEE, 2018. p. 1-6.

[10] big.LITTLE - Arm. Available online: https://www.arm.com/why-arm/technologies/big-little. Accessed by April the 10th 2020.