Computer and Information Science; Vol. 13, No. 3; 2020
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Experimental Instructional Design of Code Slicing and Data Verification by Artificial Interruption

Guoguan WEN[1], Mingliang ZHANG[1], & Qingping DOU[1]

[1] School of Intelligent Systems Science and Engineering, Jinan University (Zhuhai Campus), Zhuhai 519070, China

Correspondence: Guoguan WEN, School of Intelligent Systems Science and Engineering, Jinan University (Zhuhai Campus), Zhuhai, 519070, China.

Received: April 29, 2020 Accepted: June 8, 2020 Online Published: June 23, 2020

doi:10.5539/cis.v13n3p49 URL: https://doi.org/10.5539/cis.v13n3p49

Abstract

Students run code directly without any modification when doing programming experiments. They do not care about the process and do not understand the results. It has always been the pain of the electronic information experiment teaching mentioned by. This article, combines the previous teaching experience, proposes an artificial serial port interruption in the code. The result data returned by the interrupt upload to computer or mobile phone, combined with the experimental principle to allow students to explore the relationship between input and output, so that students can fully participate in the modification of code parameters, and verify the intermediate results. Students connect to the WIFI hotspot configured in the experiment box through their mobile phones, and lunch the online commissioning APP serial port configuration interface to complete the basic serial port settings. The slave MCU needs to download the firmware code compiled in C language in advance. The code contains the serial port output function to calibrate the running position of the code and upload the specified operation results. This article briefly describes the hardware functional chart and the functional flowchart of the interrupt code; focusing on the interface design and interrupt implementation method of online commissioning APP, as well as the test environment setup and specific test results with serial port tools.

Keywords: instructional design, serial port interruption, code slicing, data verification, APP Design

1. Introduction

According to the situation of students doing experiments in the experimental class of "RFID Principles & Application Experiments", the classroom experiment time is limited. Group of two students (Neha Katira, Laurie Williams, Eric Wiebe, 2004) just completed the experiment according to the experimental steps of the experimental guide, but they cannot really understand the experimental principles and results and are not familiar with the program. Even though students have done many experiments, their practical application ability proposed by (Edward, David, 1978) has not improved significantly. A class of students is a group of different individuals. There are differences in cognition, acceptance, and hands-on practice (Shuangyan Liu, Mathieu d'Aquin, 2017). The unified experimental content cannot meet the learning needs of different student objects. In this way, some students complete the experiment in advance and do nothing in rest time. The teacher may be busy in answering questions and ignoring this part of the students in the classroom. The teacher fails to provide these students with improved or expansive experiments in time, so that they cannot be motivated to master the principle better. The students actively explore the practice and lose their enthusiasm for the experimental course, because they feel that the experiment was too simple. There are also some students, who cannot complete the experiment on time, and feel difficult in experiment. If the experiment fails, the student will feel frustrated (Moira Fallon, Zhang Jie, Eun-Joo Kim, 2011), which will hurt their enthusiasm for learning.

Combining with the problems of the above RFID experiment class, if the key steps involved in the experiment and the results of the code can reveal through code slicing proposed by (Raghavan, Susan, 2001) during the experiment, that students could split the experiment gradually, according to the prompts. The operation results and the principle can have mutual authentication, so that the students' understanding of the experiment will be greatly improved, helping cultivate the students' ability to transfer from verification experiments to design experiments. The way to achieve this is to add an interrupt to the code by artificial means, output the running position and result of the experiment code to the computer through the serial port, and connect the experimental

principle explanation and the operation result on the APP to form a closed-loop verification effect.

2. Instructional Design

In the current experimental class involving programming (Yanqing Wang, Hang Li, 2012), students in a slightly more complicated course rely on the current ready-made code. After running the code in the class, students feel that they have completed the experimental requirements. In fact, students are not clear with the working principle of the experiment. For slightly complex comprehensive experiment experimental class, students have no way to practice it and know where to modify. If there is any direct result of the modification, after completing many verification experiments, students still feel that the implementation process is a black box proposed by (John Sweller, 1999). This deep-rooted learning method for students, and the teaching method that the experiment class is to run the code automatically, is the difficult problem that the current experiment class needs to face and crack directly.

2.1 Hardware Structure Diagram

For better to explain the following instructional design, this article makes a brief introduction to the hardware structure of the experiment box. The experiment kit is mainly composed of four modules, including MCU, serial port, WIFI module and low-frequency module. The MCU can directly control the low-frequency module; three data connection modes realize through the three-level switch: (1) The WIFI module directly connects to the MCU to realize the WIFI serial port function; (2) The serial port is directly connected to the WIFI module, which is used for the initial configuration of the WIFI module; (3) The serial port is directly connected to the MCU, which is used to download the firmware code to the MCU through the serial port. The MCU needs to implement the operation of the low frequency card according to the EM4095 chip protocol written by (EM, 2002). The specific operation content determined by operation instruction, which is sent to the MCU by the serial port (Manisha Sharma, Nidhi Agarwal, SRN Reddy, 2015). The key experimental content of the students is to understand the MCU implementation process. This course selects C language as the underlying MCU firmware development language.

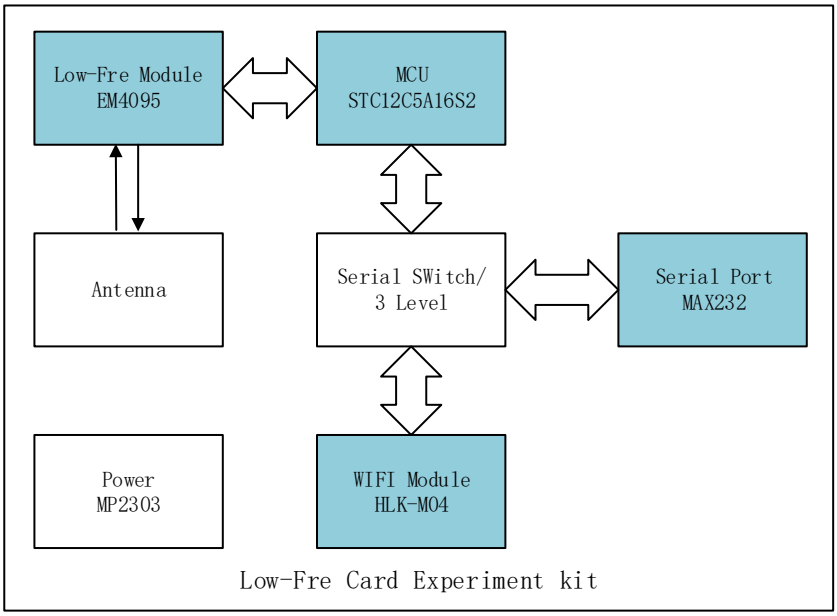

Figure 1. Hardware schematic diagram of low-frequency card experiment kit based on serial communication

The configuration of the development board includes three aspects. The first step, students use the MAX232 serial port to connect to the microcontroller directly. The host computer writes the *.hex file to the MCU through the firmware-programming tool provided by the manufacturer with USB-UART port (Silicon Lab, 2014). The second step, student uses the MAX232 serial port to connect to the HLK-M04 WIFI module referenced (Nordic Semiconductor, 2010) directly. The host computer uses the serial port to initialize and configure the module, so that the host computer can connect to the hot spot normally. The third step, the WIFI module connects to the MCU directly using the three-position switch, so that the host computer finally communicates with the slave machine through the WIFI serial port.

2.2 Functional Block Diagram of Firmware Code

To solve the problem mentioned above, this article combines the previous teaching experience and the student's feedback on the learning difficulties, and proposes a way to artificially add interrupts in the code and upload the result data to a computer or mobile phone. Based on verification method by (Jan Nossen, Richard H. Shea, 2001), program waits until the student understands this step, and then execute to next step. The functional flow shows in Figure 2.

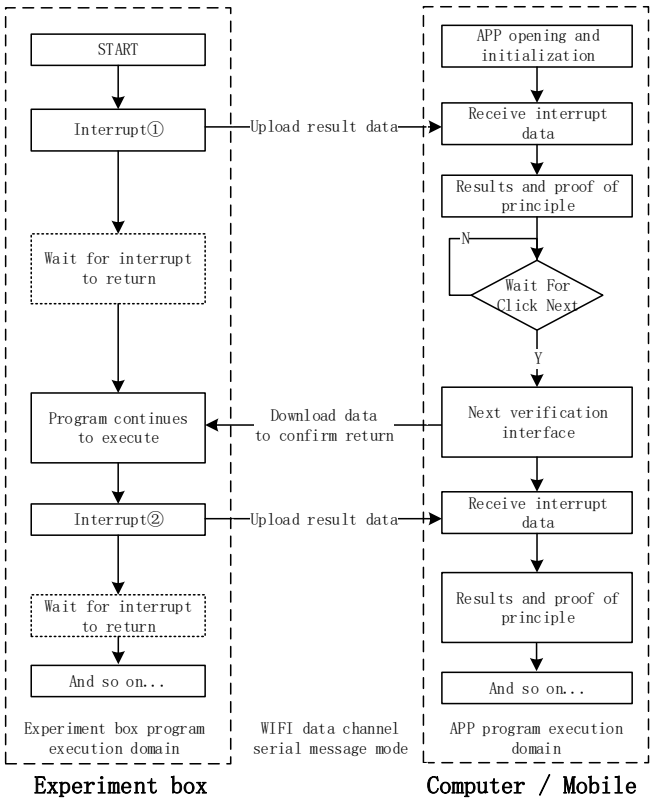

Figure 2. System functional block diagram of code process decomposition and data interaction verification

Designer inserts the serial port output interrupt in the existing code, so that the host computer can receive the data output by the serial port. At this time, the slave machine program stops executing until the upper computer confirms that the interrupt returns, and the slave machine brings the parameters that need to be called back into the actual register.

The lecturer needs to focus on solving two problems: 1. Where to add interrupts in the code and which data extracted to help explain the experimental principle; 2. How to design the interface so that the explanation of the principle and the verification of the data can be easily understood and absorbed by students. In addition, the user-friendly and convenient interface is a key factor for students' willingness to use. In this way, the students can run the code in steps. The key results in the experiment is sent to the students in a timely manner, making the students' learning process more transparent, and the students learn more calmly and confidently in the practical teaching.

2.3 RFID Online Commissioning APP Interface Design

Aiming at the second problem of interactive interface design, this article provides a usable solution. The specific design shown in Figure 3. The APP needs to lunch the interface for configuring WIFI serial port parameters at the beginning, and the specific interface and operation process will not repeated here. After the mobile APP successfully connects to the experiment box, the top of the page requires students to choose which card experiment, including three types of low-frequency card, high-frequency card and NFC card. After selecting the card type, the interface will make a simple statement according to the content of the key agreement that the teacher wants the students to master. Followed by the HEX instruction input window, you need to click the [Send] button, after entering the operation instruction. The data uploaded by the serial port of the slave machine is

output in the interrupt prompt. The actual output window will do a secondary analysis of the output content, such as printing the ID card number directly, or it may be an intermediate result of a preset register. The parameter callback window will prompt the specific register or parameter name that be called back. If there is no callback parameter, it will be blank. Students can enter a guess value according to their own understanding, or a theoretical value calculated according to the principle. After entering, click [Confirm]. The parameter value will return to the target register where the current breakpoint is located, which is equivalent to manually modifying the intermediate operation value through the window. In order to distinguish it from the input of the instruction code, the returned data is marked with an asterisk *, followed by a byte of valid data. Click [Next] to continue to the next artificial breakpoint, otherwise, the slave machine program will keep waiting in place. The APP interface will show the explanation of the next principle. The overall function is the same as above. [Previous] button only provides the interface review function, the interface temporarily stored in the previous step is directly displayed, and the slave machine will not have any operation.

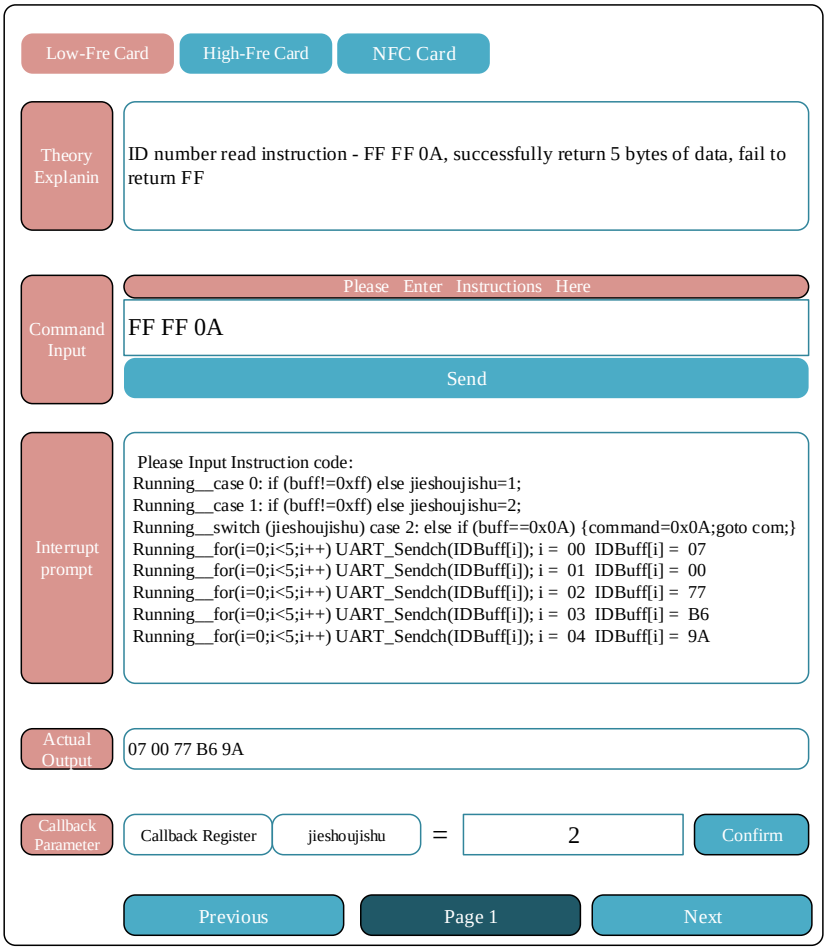

Figure 3. Mobile interface design integrating command input, data return and parameter callback functions

At the initial stage of instructional design development, the teacher should do functional plan and design according to the teaching experience and the principle knowledge required for the subsequent design experiments. Based on the learning results, we need to recollect the required knowledge reserve, and then design the software functions according to the required knowledge reserve, instead of adding interrupts for each line of code. Therefore, the students can complete the necessary proof of principle in limited classroom time. Therefore, at the beginning of the teaching design, the teacher is very critical to define the functional requirements of the software.

3. Scheme Implementation and Testing

This chapter will test the content other than the APP design. The development environment configuration includes WIFI initial configuration, preset MCU code modification, compilation and programming; the test

results based on the desktop computer as the host computer, and the WIFI communication method realizes serial reporting upload of text.

3.1 Environment Configuration and Design Realization

Taking the low-frequency card as an example, the experimental version includes the STC12C5A16S2 microcontroller, UART WIFI module, low-frequency read-write chip and corresponding antenna winding. In order for the microcontroller to communicate directly with the WIFI serial port, the UART WIFI module needs to configure with the serial port configuration tool. Including network name, encryption method and initialization password, port number 4001, baud rate 9600, data bit 8, verification code NONE, stop bit 1, local IP 192.168.0.162, mask 255.255.255.0. Finally, the host computer receives the WIFI link FRO_RFID hotspot, and the host computer can transmit the ASCII serial message through the WIFI channel to the MCU with socket format (Yan-Rong Tong, Zuo-Bin Li, 2017). The software functions and principles are similar to the serial assistant.

After completing the above configuration, instructor insert the serial port interrupt print string in the corresponding key code line, according to the teaching difficulties in the experimental course. The string mainly displays the currently executed code line and related register values. At the line of code, the function Uart1_String will generate a serial port interrupt and print out the specified content. For example, we add code in the command header FF FF recognition with

```
if (buff!=0xff) {jieshoujishu=0; Uart1_String("\r\nRunning__case 0: if (buff!=0xff) jieshoujishu=0;");}
else {jieshoujishu=1; Uart1_String("\r\nRunning__case 0: if (buff!=0xff) else jieshoujishu=1;");}
```

With two lines of serial output statements, students can know how the code is executed after entering the command header. For another example, we add it in the FOR loop that reads the ID card number with

```
Uart1_String("\r\nRunning__for(i=0;i<5;i++)UART_Sendch(IDBuff[i]); i =");UART_SendchASC(i);Uart1_String("IDBuff[i] = ");UART_SendchASC(IDBuff[i]);
```

Students will know which register and which code of the five-byte ID card number are output in bytes.

3.2 Test Results

As shown in Figure 4. The TCP serial port application of the host computer can directly output the string specified by the Uart1_String function preset in the code, prompting the current code running position and the register i value, and its corresponding IDBuff [i]. Students input FF FF 0A through the serial port tool or RFID online commissioning APP, the serial port output window will display the following content, of which 07 00 77 B6 9A in hex is the ID card number.

Turned on slave machine, and the serial port outputs a prompt message:

Please Input Instruction code, If want to Interrupt or Continue, Input Symbol # to return!

Please Input Data with Pre-Symbol *(or 2A) value (HEX) for jieshoujishu.

The first stage test input the ID card operation instruction “FF FF”, the slave machine has no operation and no output prompt, indicating that the slave machine is waiting for execution. The *second* stage test, Enter the character # according to the prompt, the slave machine prompts that the character successfully received with “**Symbol: # is Received!**” and the program can continue to execute. The *third* stage test, input the first FF, the slave machine returns code instructions and jieshoujishu=1 with “**Running__case 0: if (buff!=0xff) else jieshoujishu=1;**”. Enter the second FF, the slave machine returns code instructions, and jieshoujishu=2 with “**Running__case 1: if (buff!=0xff) else jieshoujishu=2;**”; Enter the read ID card number instruction code 0A, the slave machine returns code instructions and return the intermediate result i and IDBuff [i]. After five time FOR loops, we get the ID card number. The *fourth* stage test, the value of jieshoujishu directly modified without input “FF FF”. Send * as preamble, the slave machine prompts that the character received with “**Symbol: * is Received!**” Send 02H, the slave machine prompts jieshoujishu = 02H write success with “**Input data for 'jieshoujishu' make sense with value (HEX): 02**” execute the operation command 0A, the salve machine returns the ID card number; it means that the value of the register is valid by directly modifying. The test results are consistent with the second stage. In the *fifth* stage test, stopping the program by entering # make slave machine actually waiting in place. Enter #, the slave machine prompts that the character received with “**Symbol: # is Received!**” Enter FF for twice, the slave machine has no operation and no feedback, indicating that the program has stopped and is consistent with the test in first stage.

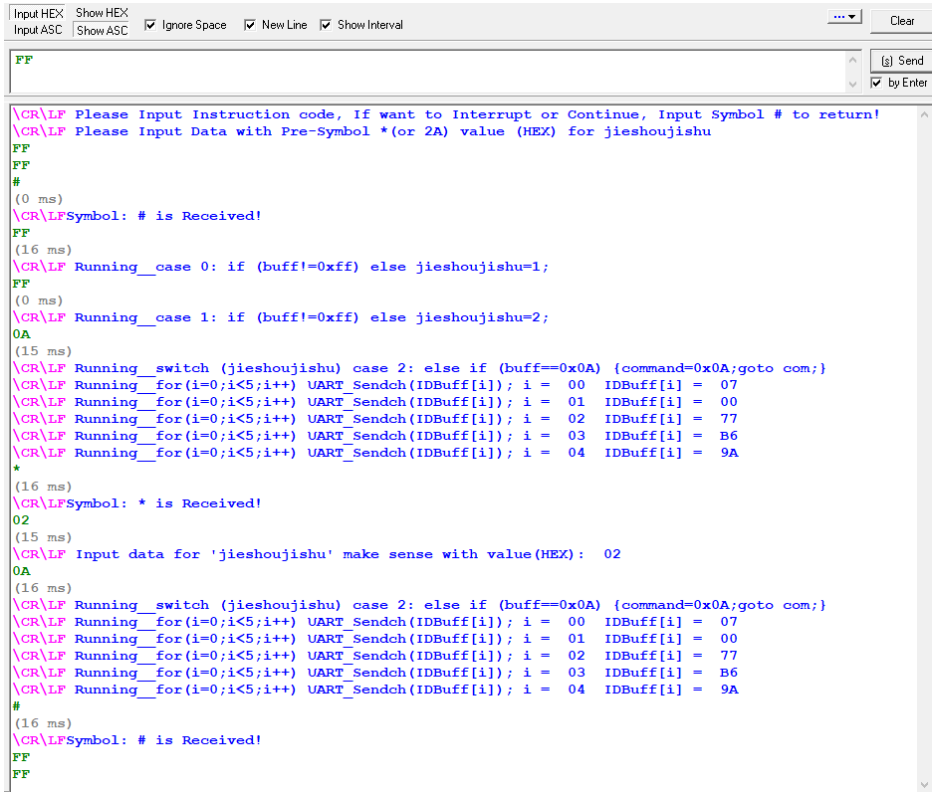

Figure 4. Command input and interrupt output test results under serial port application

The test completed the command input and return, code slicing prompt, return intermediate calculation result, manual modification of register intermediate result, manual stop or continue execution through the desktop serial port assistant. It considered that the feasibility of the implementation of code slicing and data mutual verification schemes by artificial interruption, so that students can explore the relationship between code input and output in programming experiment classes.

4. Discussion

This paper only completed the serial port output test on desktop application. Because the general serial port assistant only supports HEX or ASCII display, and the serial port output itself contains both character strings and 16-bit data, so in this article, 16-bit data convert to ASCII before the data is output. So that the host computer application only needs to output the display ASCII string. As long as the operation of the salve machine is involved, the host computer sends operation data to the salve machine through the WIFI serial port. According to the original code, only ID card operation instruction input supported, and other instructions or data are invalid. The mobile terminal APP interface has two more operation for the slave machine: "Parameter Callback" and "Next Step". In parameter callback, once the value of the callback parameter is determined and press [Confirm], the APP first sends the preamble *****. Then send the specific value. [Next] button used to stop interruption, let the program continue to execute, APP sends the control character # to slave machine. If you want to stop the execution, you can send # again.

When the student completes the verification experiment, the teacher will propose a design experiment. For example, add the ID card number just read to add 1 as the new card number and rewrite it to the original low frequency card. Students will quickly know that the card number comes from the register array IDbuff[] through the code line prompting above, and know how to modify the code to satisfy the design requirements during the design experiment.

Using this method, students can clearly know how their input instruction be executed and what results will be obtained. Students understand the code more intuitively, and the teacher does not need to explain the implementation method too thoroughly before the experiment class. The students can use the teacher preset code explores the relationship between input and output in sections. This article takes the reading of the ID card

number in the low-frequency card experiment as an example. In practice, much complicated coding processes are involved in the teaching. Students are generally reluctant to understand in depth how the code realizes this coding gradually. Only by allowing students to understand passively can students understand the code more deeply.

5. Conclusion

Students only run code when doing experiments; they do not care about the process, and do not understand the results. This problem has always been the pain of the electronic information class. This paper bases on the hardware and software support provided by the existing RFID experiment suite. Teachers need to modify the code according to the teaching needs, and add interrupt data collection points without changing the protocol specifications and functions. When students use the host computer, code execution stays at the interruption position, until the students understand the data values returned by the current device and verify the principle with the correct. Students then click to continue execution, so that students can fully participate in code parameter modification and verification of process results, which is conducive to students having a deeper understanding of principles and codes. The experiment teaching changed from running the code after the completion of the task to verifying while running the code. After the students modified it, they can immediately find whether the result is correct. This is a slow process of absorption and reception, which effectively resolves the pain and difficulties of the above-mentioned experimental classes, and provides a good example of teaching reform for the subsequent different courses. For example, the "C Language" (Leonard J. Mselle, 2011), "Principle of Single Chip Microcomputer", "Embedded Development", and "Sensor Network".et experiment classes all require students to compile the code to complete the expected demand of the class experiment course. The students initially execute the program according to the original demo, and then according to the difference between the current experimental result and the experimental requirements preset by the teacher, students actively modify part of the code. For high-level programming experiment, classes such as "Embedded Development" and "Sensor Networks", students find it difficult to understand the code, because the amount of code in an experiment is more than a few hundred lines. The original intention of this article is to let students know which statements in the current code are the most important. Students need to pay attention and understand. Otherwise, students will have difficulty meeting the preset design experiment requirements (Agostino Marengo, Alessandro Pagano, Alessio Barbone, 2012). Compared with traditional online commissioning & simulation, the instructors focus their teaching on the line of code to allow students to learn more targeted. The combination of teaching experience and teaching methods will make students more profitable.

Acknowledgments

The authors acknowledge financial support from the 'Fundamental Research Funds for the Central Universities' of JNU with Grant No. 11617361.

References

Agostino, M., Alessandro, P., & Alessio, B. (2012). *Adaptive Learning: a New Approach in Student Modeling.* The ITI 2012 34th Int. Conf.on Information Technology Interfaces, June25-28, 2012. PP217.

Edward, E. G., & David, L. A. (1978). Levels of Instructional Difficulty as Related to On-Task Behavior, Task Completion, and Comprehension. *Journal of Learning, Disabilities, 11*(9). https://doi.org/10.1177/002221947801100905.

EM MICROELECTRONIC-MARIN SA, EM 4095 Read/Write analog front end for 125kHz RFID Basestation. Retrieved from www.emmicroelectronic.com.2002

Jan, N., Richard, H. S., & Jon, R. (2001). *New Developments in Multiphase Flow Modelling and Field Data Verification.* Engineering Technology Conference on Energy. ETCE2001-17130, 2001, pp. 1005.

John, S. (1999). Visualisation and Instructional Design. *Australian Educational Review, 1502.*

Leonard, J. M. (2011). *Enhancing Comprehension by Using Random Access Memory (RAM) Diagrams in Teaching Programming: Class Experiment.* In Proceedings of the 23rd Annual Workshop of the Psychology of Programming Interest Group.

Liu, S. Y., & Mathieu, A. (2017). *Unsupervised Learning for Understanding Student Achievement in a Distance Learning Setting.* 2017 IEEE Global Engineering Education Conference, pp. 1373. https://doi.org/10.1109/EDUCON.2017.7943026

Manisha, S., Nidhi, A., & SRN, R. (2015). *Design and development of daughter board for USB-UART communication between Raspberry Pi and PC.* International Conference on Computing, Communication &

Automation, pp. 944-945.

Moira, F., Zhang, J., & Eun-Joo, K., (2011). Using Course Assessments to Train Teachers in Functional Behavior Assessment and Behavioral Intervention Plan Techniques (2011). *Education and Human Development Faculty Publications, 6,* 51. Retrieved from https://digitalcommons.brockport.edu/ehd_facpub/6

Neha, K., Laurie, W., Eric, W., Carol, M., Suzanne, B., & Ed Gehringer. (2004). *On Understanding Compatibility of Student Pair Programmers.* The 35th SIGCSE technical symposium on Computer science education, March 2004 pp. 7-11. https://doi.org/10.1145/971300.971307

Nordic Semiconductor. (2010). *nRF24LE1-Ultra-low Power Wireless System On-Chip Solution.* Preliminary Product Product Specification v1.6.

Raghavan, K., & Susan, H. (2001). Using Slicing to Identify Duplication in Source Code. *International Static Analysis Symposium. SAS 2001, Static Analysis,* pp. 40-56. https://doi.org/10.1007/3-540-47764-0_3

Silicon Labs. "USB to UART Bridge | Silicon Labs", 2014. Retrieved from http://www.silabs.com/products/interface/usbtouart/Pages/usb-to-uart-bridge.aspx?tab=info

Tong, Y. R., & Li, Z. B. (2017). *Design of Intelligent Socket Based on WiFi.* 2017 4th International Conference on Information Science and Control Engineering (ICISCE). 21-23 July 2017, pp. 952. https://doi.org/10.1109/ICISCE.2017.201

Wang, Y. Q., Li, H., Feng, Y. Q., Jiang, Y., & Liu, Y. (2012). Assessment of programming language learning based on peer code review model: Implementation and experience report. *Computers & Education, 59,* 415-416. https://doi.org/10.1016/j.compedu.2012.01.007