Low-Cost and Effective Student-Orientation Mastery Learning Simulator for Programming Course

Hsin-Hsiung Huang, Juing-Huei Su, and Chyi-Shyong Lee

***Abstract*—The training method, one teacher teaches one student to solve his programming mistakes by using one computer, costs a little time. In this study, the low-cost and effective modeling, which the teacher integrates the common mistakes by students into the programming simulator, is presented. In programming course, the basic and common mistakes made by the students are first listed and then the teacher takes these mistakes as the course objective of mastery learning. For each course objective, the corresponding mistakes are randomly generated by the programming simulator. Hence, the students can generate the wrong source codes and then finish the exercises to fix all mistakes by themselves. The wrong source codes are quite different when the students perform the programming simulator. Therefore, the students need to realize the programming principle instead of memorize the answers. Experimental results show that the students indeed master to fix these mistakes which are corresponding to the objectives.**

***Index Terms*—Programming simulator, mastery learning, randomly generate, common mistakes.**

I. Introduction

Programming skills are getting more and more important today. Many researches focus on how to improve the students' performance in programming and industrial skills [1]-[5]. The famous teacher provides the "master learning" concept to the teacher to define the content and skills, which the teachers wanted students to learn. When students learn the content of one unit and the following unit goes later [6]. Bloom presented the concept of combination of the mastery learning with other useful method may to help students to learn better [7]. The study guided the teachers to think how to search the good methods to train the students. Information of the web site is the mastery learning approaches [8]. The important paragraph illustrates that mastery programs are effective to the students [9]. The authors provided the graphic user interface-based system to select test items [10].

The contributions of this study are as follows. First, the implementation of the programming simulator, which considered the objectives and the corresponding common mistakes, is presented to use in the programming course, such as data structure. It means that the programming simulator can be adjusted when the teachers modify the course objectives according to the students' demands. Second, the programming skills of students are enhanced because the students can try to fix the different wrong source codes, which are randomly generated from the programming simulator. For each exercise with wrong codes, students must finish 100% mistakes until source code without mistakes can pass the complier. Students can be confident to programming skills after several practices.

The organization of this study is as follows. Section II describes the motivation to involve the programming simulator for mastery learning. Section III discussed mastery learning method, which includes the course planning, the programming simulator, and mastery learning. Section IV illustrates results by utilizing the programming simulator into the course. Finally, some conclusions are made in Section V.

Manuscript received October 16, 2015; revised December 23, 2015.

Hsin-Hsiung Huang, Juing-Huei Su, and Chyi-Shyong Lee are with the Lunghwa University of Science and Technology, Taoyuan, 33306, Taiwan (e-mail: pp022@mail.lhu.edu.tw).

II. Motivation

The students of University of Science and Technology are usually lack of motivation to actively learn the programming skills. Some students cannot buy the expressive training tools kits to enhance the programming skills. How can we develop the programming simulator with low-cost and easy-to-use?

A. Integrate the Programming Simulator into Data Structure

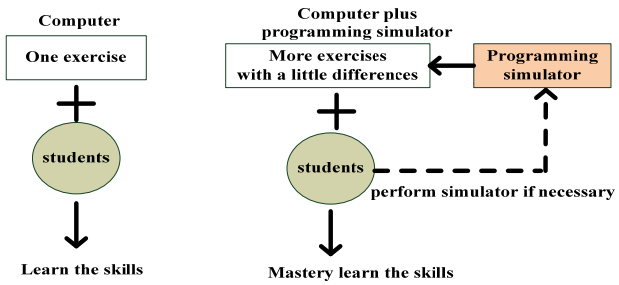
Mastery learning, which focuses on certain topics to train students, is useful to the programming-related courses. The programming skills, which should be developed for al long time with many practices, are very important and basic to the students of the electronic engineering departments. Therefore, the course designer plans to integrate the mastery learning into the programming-related courses, such as data structure. For the exercises in data structure, some students cannot finish the source code because there are some mistakes. The teacher should spend a little time to help all students to finish their exercises. Students do not learn very well because they do not practice the source codes many times. By the observation, the teacher first finds the common mistakes, which many students may make. Then the programming simulator, which integrates these mistakes, is developed by the teacher. During the course, the teachers observe students' outcomes and determine practice times to the programming simulator. Summary, this programming simulator is low-cost but effective because the simulator is designed by the teachers according the programming mistakes in this course.

III. Method

In this section, course planning which the teacher observes the common mistakes from the students, the learning sheet which contains a little of operations of the programming simulator by students in physical classroom, and mastery learning to help the students, are as follows.

A. Course Planning

Originally, the teacher first finds the learning objectives after observing the students. Then the teacher implements the programming simulator to train students. In the physical course, students are merged into several groups. The students generate the exercises with some mistakes and then the students try to fix these mistakes. After several testing times, students integrate these data into the recording file and upload them to the school server.



(a) course without master learning (b) course with master learning

Fig. 1. Illustration of the programming simulator for mastery learning.

TABLE I: OBJECTIVES OF THE PROGRAMMING SIMULATOR

	Objectives to mastery learning
1	Know the operation of mastery learning.
2	Learn the use of symbols, such as double quotes, comma, and semicolon, in the source codes.
3	Apply the use of sub-routine, for loops and variable definition.

Practically, the objectives, which teachers want to train the students, are first defined. In Table I, three objectives are found to train the students in programming skills. In Table II, the mistakes, which are corresponding to the objectives, are briefly shown. The wrong examples are listed in the right column. In Table III, the suggested answers are shown and help students to finish the exercises (i.e. pass the complier) [11]. Of course, the other answers, which can pass the complier, are also allowed. For example of the mistake which is labeled 2, the suggested answer is printf("\n\n");. The other answers, such as printf("\n"); or printf("\n\n\n"); are also right answers. The standard is that students could pass the complier because students are asked to capture the successful message and paste them in the learning sheet which is shown in Fig. 2.

B. Learning Sheet for Mastery Learning

B.S. Bloom [1] mentions that the individual differences in learners should be considered in the physical course. In this programming course, the practice time (defined as t) is set to be 3. It means that the three (t=3) practice times are enough to learn the course objective. Hence, the learning sheet used in the exercise is shown in Fig. 2(a). In the first exercise, students should copy the wrong code in the Exercise 1-1. After passing the complier [11], students should copy the right codes to Exercise 1-2. Besides, the successful screen must be also listed the Exercise 1-3. The similar operations are used from Exercise 2-1 to Exercise 3-3. Therefore, by using the programming simulator containing the course objectives, the students in each group can be mastery learning for the objectives in Table I after fixing the all mistakes (like Table II and Table III).

TABLE II: LIST OF THE MISTAKES GENERATING TO THE WRONG CODE FOR EACH OBJECTIVE

	Description of mistakes	Wrong examples
1	Mistakes of sub-routine, for loops and variable definition.	ffor(innt i=200; i<+1000; i=i+=100)
2	Mistake of double quotes.	printf(\n\n);
3	Mistake of semicolon.	total = total -now.

4	Mistakes of semicolon, colon, and comma.	else if (computer = you);
		else if (computer = you):
		else if (computer = you).

5	Other errors.	scanf(%f,@ans)

TABLE III: SUGGESTED ANSWERS TO THE WRONG CODE

	Wrong examples	Suggested answers
1	ffor(innt i=200; i<+1000; i=i+=100)	for(int i=200; i<+1000; i=i+100)
2	printf(\n\n);	printf("\n\n");
3	total = total -now.	total = total –now;

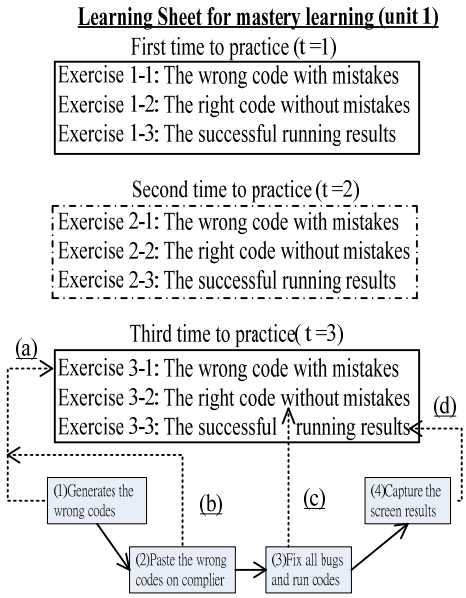
4	else if (computer = you);
	else if (computer !- you):
	else if (computer ^=you).
		else if (computer == you)
		else if (computer == you)
		else if (computer == you)

5	scanf(%f,@ans)	scanf("%d",&ans)

Learning Sheet for mastery learning (unit 1)

Exercise 1-1: The wrong code with mistakes
Exercise 1-2: The right code without mistakes
Exercise 1-3: The successful running results
Exercise 2-1: The wrong code with mistakes
Exercise 2-2: The right code without mistakes
Exercise 2-3: The successful running results
Exercise 3-1: The wrong code with mistakes
Exercise 3-2: The right code without mistakes
Exercise 3-3: The successful running results

(a) original learning sheet assigned to students



(b) more descriptions for learning sheet

Fig. 2. Learning sheet for mastery learning.

Fig. 2(b) shows the more descriptions for learning sheet. According to the above discussion, the practice time is set to be three times. In each practice time, the tasks are to finish three exercises. For example, the first time to practice (t=1) is to finish Exercises 1-1, 1-2 and 1-3. Each practice contains four steps, including 1) generate the wrong code by using the programming simulator; 2) paste the wrong code on the free complier Dev C++[11]; 3) fix all bugs which are randomly generated in step (1) and the students can run the codes after fixing these bugs; and (4) capture the successful screen when the programming code performs without any bugs. Moreover, the four dotted lines, which connect the operation steps and the corresponding exercises, are used to show the relationship between the students' operations and the learning sheet. For example, the dotted line, which is marked (a), denotes the connection from step 1 to Exercise 3-1. Similarly, the dotted line, which is marked (b), shows the connection from step 2 to Exercise 3-1. Hence, the students finish Exercise 3-1 after making the steps 1 and 2. Exercises 3-2 and 3-3 are finished after the students making the steps 3 and 4, respectively.

According to above descriptions, the learning sheet is designed to guide the students how to make several practices, see Fig. 2. In fact, the format of the learning sheet can be adjusted according to the training objectives. Fig. 2(b) shows that each practice (t=1, 2 or 3) has four steps and the students can sequentially record their wrong code with mistakes, their right code without mistakes and the successful screen results. For the objectives that are made by the teachers, the students can develop and speed up their programming skills after many practice times.

C. Design and Implement the Programming Simulator

To enhance the programming skills, the programming simulator, which is discussed in this section in detail, provides the exercises to make students learn-by-doing [12]. The teaching concept is that much practice for the similar exercises with wrong code makes the students learn the programming skills better.

The outline of the programming simulation is shown in Fig. 3. First, the students' demands are used to set the course objectives, see TABLE II and Fig. 4. Second, for each objective, the teachers' defined mistakes to make students learn the use of codes, see TABLE II. In TABLES II and III, the mistakes are just taken for example. In fact the more mistakes can be designed to help the students. Third, the pre-defined mistakes are randomly generated and the mistakes are similar but a little different. Finally, all mistakes are recorded and integrated into a test file with the mistakes, see Fig. 5.

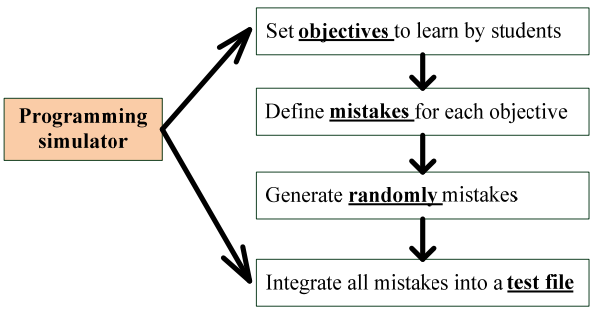


Fig. 3. Illustrate the programming simulator.

D. Mastery Learning Procedure

Before the course, the teaching activities of this course are as following:

1) Teacher observes the students and finds the mistakes to determine the learning g objectives, see Table I. In fact, the number of g is suggested to be designed according to the background and the learning motivations of the students. Too many objectives makes the test files larger. Therefore, the students should spend a little time to fix these mistakes (g=3 in this study).
2) The programming simulator, which makes from $g1$ to $g2$ mistakes for each objective, is designed to train students, where $g1$ and $g2$ are the lower and upper times for each objective, respectively.

In the course, students are divided into a set of groups to finish the programming exercises that are generated by the programming simulator. This procedure is as follows:

(P.1) Students of each group should download the learning sheets (Fig. 2) and the programming simulator (Fig. 3).

(P.2) In the first time, the exercise with wrong mistakes is generated and is pasted on the C language complier [11].

(P.3) Students try to fix these mistakes by discussing with the students and the teacher, see test file in Fig. 6.

(P.4) In the second time, run steps P.2 to P.3;

(P.5) In the third time, run steps P.2 to P.3;

(P.6) Students integrate the results into the learning sheet (Fig. 2) and upload the learning sheet (Fig. 2) to the school server.

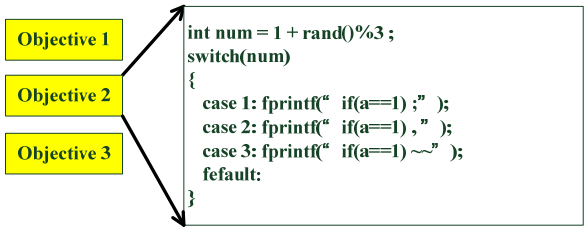


Fig. 4. Illustrate the relationship between the objectives and mistakes.

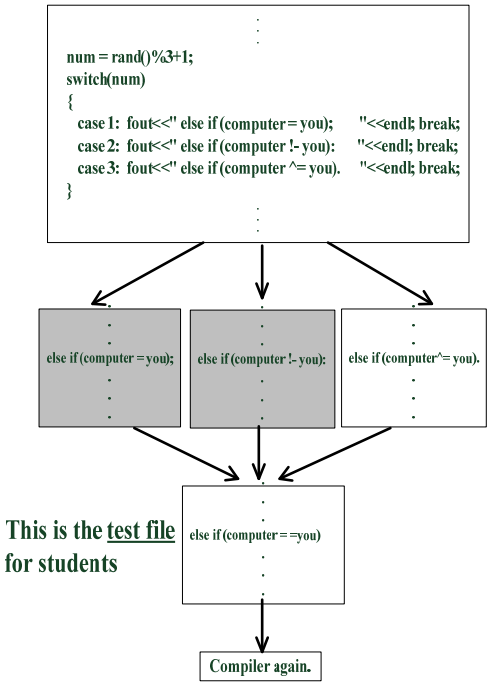


Fig. 5. Illustration of the programming simulator randomly generates and integrates mistakes into a test file for students.

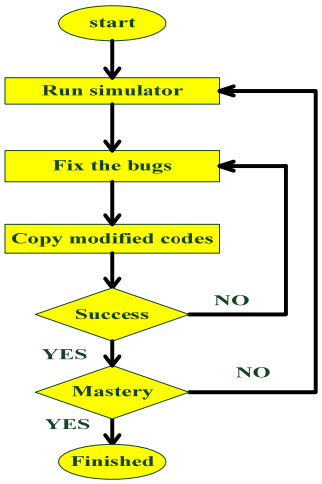


Fig. 6. Illustration of the mastery learning procedure.

E. The Physical Course of Mastery Learning

A simple example is used to illustrate how the programming simulator is used for mastery learning, see Fig. 7 and Fig. 8. First, the pre-defined three codes are generated randomly because the function of the "rand()%3+1" produces the numbers of 1,2, and 3, respectively. The wrong code with mistakes is generated by using the programming simulator, see Fig. 3, Fig. 4 and Fig. 5. Third, the students in each group may fix the different wrong codes, see Fig. 6 and Fig. 7. The students are not boring and know the programming skills for different typos. The operation of the physical course is illustrated in Fig. 8. Students 1 and 2, which are group into a team, are discussed to each other (Step 1 in Fig. 8). When they meet the serious mistake, they can ask the teacher for help (Step 2 in Fig. 8). Finally, they try to fix all mistakes (Step 3 in Fig. 8).

IV. Results

In this study, the course objective is set to be 3 (g=3, see Table I) and makes students learn the following skills:

(Goal 1) Know the operation of mastery learning.

(Goal 2) Learn to use the symbols, such as double quotes, comma, and semicolon, in the source codes.

(Goal 3) Apply the use of sub-routine, for loops and variable definition.

In this course, there are 55 students to learn the data structure. Students are divided into groups. The questionnaires are from 53 students to summarize their opinions about this programming simulator. The teacher observes the content is small size programming exercise and the practice time is set to 3(i.e. M=3) to achieve the mastery learning. Actually, the practice time is determined according to the number of objectives, the background of the students and the difficulty of the programming exercises. According to the computer equipments with the programming simulators, at most P (P is set to 2 in the study) students are merged into a group.

Table III is the questionnaire from the students. The results of the questionnaire are most positive and most of them agree that the programming simulator is effective to enhance the programming skills. Item 1 denotes that students agree the arrangement for the additional programming simulator. Fig. 9 illustrates the distribution of points 1,2,3,4 and 5. Item 2 show that students know mastery learning is to learn the one topic iteratively. Items 3 explores if students like the programming simulator of the simulator. The result is shown in Table III and Fig. 10. The course objective is to train the ability to place the comma. Item 5 shows the course objective are good because the results of "I learn whether to place symbol "comma" or not for the source codes" is 4.25 and the result is quite positive. Summary, the questionnaire distribution is shown in Fig. 11. For all items, students assign the high points. It means that most responses from students are quite positive.

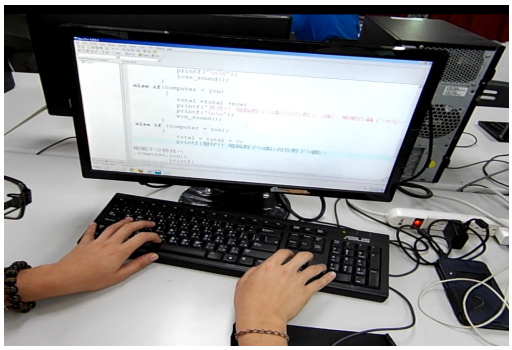


Fig. 7. Illustration of master learning by using the programming simulator.

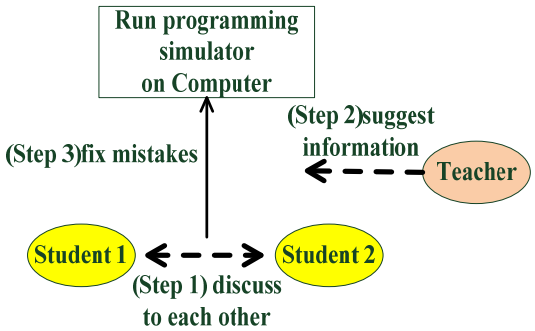


Fig. 8. Illustration of the mastery learning procedure discussing with group members or the teacher.

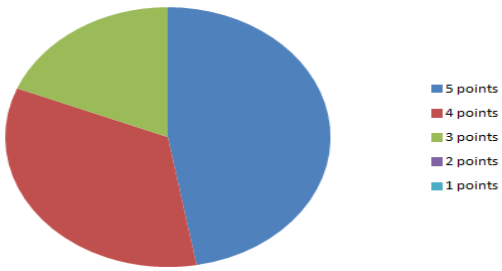


Fig. 9. Illustration of distribution from 5 point to 1 point for Item 1 of the questionnaires.

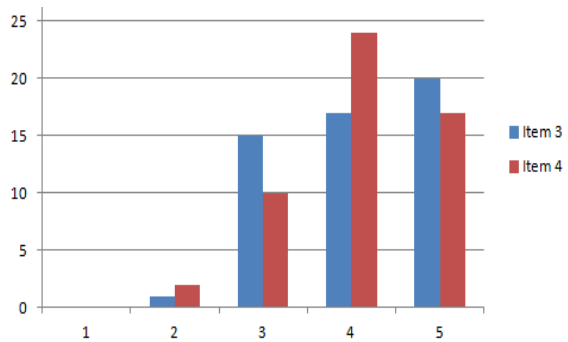


Fig. 10. Illustration of distribution from 5 point to 1 point for Item 1 and 2 with lower points.

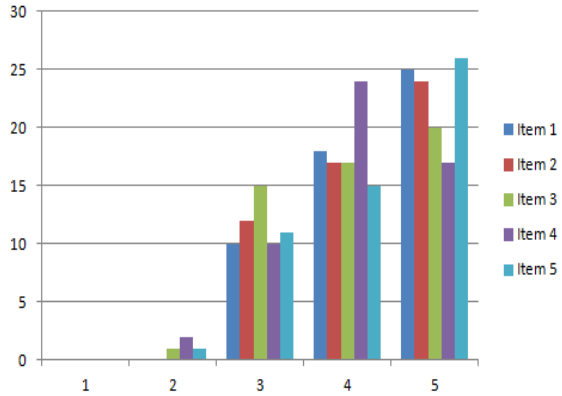


Fig. 11. Illustration of distribution from 5 point to 1 point for all Items 1 to 5.

TABLE III: QUESTIONNAIRE FROM THE STUDENTS

Item	Questionnaire content	Points (1~5 points)
1	I agree that the arrangement for the additional programming simulator.	**4.28**
2	I know that mastery learning is to learn the one topic iteratively.	**4.23**
3	I like the programming simulator to enhance the programming skills.	**4.06**
4	I develop the confidence with many practices.	**4.06**
5	I learn whether to place symbol "comma" or not for the source codes.	**4.25**

V. CONCLUSION

The low-cost and effective learning modeling is developed and utilized into the programming course. According to the course objectives, the common mistakes of the programming skills are considered and integrated into the programming simulator. Each time, the wrong source codes are randomly generated and the students try to fix these mistakes until all mistakes are solved. The learning sheet is given to help the students to record the wrong codes with mistakes, right codes without mistakes and the successful results. When the students finish these exercises, which are assigned by the teachers, the students may fix the similar bugs more efficiently. Experimental results show that the students like to use the programming simulator because they learn to fix the common mistakes by iteratively utilizing the programming simulator.

ACKNOWLEDGEMENT

The authors would like to thank the financial support from Lunghwa University of Science and Technology to develop the new technical approaches to the students of the department of Electronic Engineering of Lunghwa University of Science and Technology.

REFERENCES

[1] L. Jing, Z. X. Cheng, J. B. Wang, and Y. H. Zhou, "A spiral step-by-step educational method for cultivating competent embedded system engineers to meet industry demands," *IEEE Transactions on Education*, pp.356-365, 2011.

[2] K. Belghith, R. Nkambou, F. Kabanza, and L. Hartman, "an intelligent simulator for telerobotics training," *IEEE Transactions on Learning Technologies*, pp. 11-19, 2012.

[3] A. G. Ramos, M. P. Lopes, and P. S. Avila, "Development of a platform for lean manufacturing simulation games," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, pp. 184-190, 2013.

[4] L. S. Myneni, N. H. Narayanan, S. Rebello, A. Rouinfar, and S. Pumtambekar, "an interactive and intelligent learning system for physics education," *IEEE Transactions on Learning Technologies,* pp. 228-239, 2013.

[5] D. W. Parent, "Improvements to an electrical engineering skill audit exam to improve student mastery of core EE concepts," *IEEE Transactions on Education,* pp. 184-187, 2011.

[6] B. S. Bloom, "Learning for mastery," *Evaluation Comment (UCLA-CSIEP)*, pp. 1-12, 1968.

[7] B. S. Bloom, "The search for methods of group instruction as effective as one-to-one tutoring,"*Educational Leadership*, pp. 4–17, 1984.

[8] Mastery_learning. [Online]. Available: https://en.wikipedia.org/wiki/Mastery_learning

[9] C. L. Kulik, J. A. Kulik, and J. Bangert-Drowns, "Effectiveness of mastery learning programs: A meta-analysis," *Educational Research,* pp. 265-299, 1990.

[10] Y.-C. Lin, T.-M. Hsieh, H.-O. Hsieh, Y.-S. Hung, and W.-L. Chen, "Test item selection system," *GCCCE2000*, 2000.

[11] Devcpp. [Online]. Available: http://www.math.ncu.edu.tw/~jovice/c++/boards/devcpp.htm

[12] Wiki. (2015). [Online]. Available: https://zh.wikipedia.org/wiki/%E7%BA%A6%E7%BF%B0%C2%B7%E6%9D%9C%E5%A8%81

Hsin-Hsiung Huang becomes an IEEE member since 2010 and was born in Taiwan in Oct. 1974. He received the M.S. and Ph.D degrees in the Department of Information Computer Engineering and Institute of Electronic Engineering from Chung Yuan Christian University, Taoyuan, Taiwan, in 2000 and 2008, respectively. He is working toward the algorithm–related fields, such the applications of line-following maze robot for the shortest path problems and EDA algorithms for the VLSI, the floorplanner and performance-driven routing with the obstacles.

He is currently an associate professor in the Department of Electronic Engineering, Lunghwa University of Science and Technology in Taoyuan in Taiwan. From 2000 to 2002, He is a hardware engineering to design the Ethernet product at Accton Corporation, Hsin-Chu, Taiwan. From 2002 to 200, he serves as a research and development engineering and focus on the chip design for the 10/100/1000 Mbps Ethernet MAC at TM-Technology Corporation, Hsin-Chu, Taiwan.

Dr. Huang is also an oversea member of Institute of Electronics, Information and Communication Engineers (IEICE for short) since 2009. Some publications are listed in IEEE conferences, including ISCAS, MWSCAS and ISIC, respectively.

Juing-Huei Su was born in Tainan, Taiwan, in 1965. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 1987, 1989, 1993, respectively. From 1993 to 1995, he served as a military officer in the army. In 1995, he became a senior engineer with the Taian Electric Co., Ltd.

Since 2007, he has been a professor with the Department of Electronic Engineering, Lunghwa University of Science and Technology, Taoyuan, Taiwan. He is now interested in developing his own learning platforms for automatic control and robotic education.

His research interests also include robust control theory, power electronic systems, and embedded control system implementations.

Chyi-Shyong Lee received the B.S. from National Taipei University of Technology, Taiwan, in 1979, and the master degree from the National Tsing Hua University, Taiwan, in 1985, both in electrical engineering. From 1985 to 1988, he served as a lecturer with the Hwa Hsia Institute of Technology, Taiwan. He was a lecturer with the Department of Electronic Engineering, Lunghwa University of Science and Technology, Taiwan, from 1989 to 2007, and is now an associate professor. Currently, his research interests include digital control of power electronic systems and the applications of microcontrollers and embedded systems.