An Embedded Systems Remote Course

André Sanches Fonseca Sobrinho[1]
[1] Universidade Tecnológica Federal do Paraná (UTFPR), Cornélio Procópio, Brazil

Abstract— **This paper presents an embedded systems course offered remotely to undergraduate students of the Universidade Tecnológica Federal do Paraná (UTFPR), Cornélio Procópio campus, during COVID-19 pandemic. The course allowed undergraduate students of control and automation and computer and electronic engineering to explore the use of the real time operation system in microcontrolled systems using different free software, including one that emulates the development kit used in practical lab classes. Student feedback and course evaluation are along presented, with reflections on the differences between in-class and remote modes.**

Index Terms— **Embedded systems, microcontroller, teaching, remote classes.**

Introduction

Embedded System can be defined as a device that contains hardware and software components to perform a single function and to work with minimal or no human interaction. These systems operate in constrained environments where memory, computing power, and power supply are limited [1]. Such type of systems can be found inevery aspect of our daily lives: electronic toys, cellular phones, TVs, cars, trains, medical systems such as pace makers and ventilators, safety-critical systems such as anti-lock brakes and airbag systems and defense systems such as missile guidance [2]. The adoption of microcontrollers in embedded systems as a near universal solution in electronic design has come about in part because of their use as miniature computers, but mainly because of their low cost and rapid application development facility [3].

The Embedded Systems course is a important professional course in colleges and universities, which have strong characteristics of practicality and applicability [4], which focuses on the use of real-time operating systems (RTOS) in microcontrollers. This is verified in the pedagogical projects of the bachelor's degree in computer, electronic and control and automation engineering in most public and private universities worldwide [5-10].

At Universidade Tecnológica Federal do Paraná (UTFPR), Cornélio Procópio campus, the embedded systems presential classes in control and automation, computer and electronic engineering degrees were suspended from March 16, 2020, two weeks after the start of classes due COVID-19 pandemic. Therefore, based on experience in that campus, this paper presents in details the structure of an embedded systems course to be offered remotely using tools like Google Meet for synchronous learning, and free software such as C18 compiler [11], MPLAB Integrated Development Environment (IDE) [12], FreeRTOS [13] and PicSimLab [14] for the students performed the proposed exercises on their computers, including exercises that would be executed using development kits in practical laboratory classes. Student feedback and course evaluation are along presented, along with reflections on the differences between classroom and remote modes. It is hoped that this article can help other institutions to develop similar embedded systems courses.

The organization of the rest of the article is as follows. Second section sumarizes the structure on embedded systems in-class course and the third section presents in details how the topics of the embedded systems remote course are explored, using exercises showed in on-line theorical classes and included in the student's exercises lists. Fourfth section describes the course assessment from students and their feedback. Fifth section concludes our article.

I. EMBEDDED SYSTEMS IN-CLASS COURSE OVERVIEW

The Embedded Systems course offered every semester to Control and Automation, Electronic and Computer Engineering bachelors requires the Microcontrollers course as a prerequisite for students. The C18 compiler, the MPLAB integrated development environment and the PIC18F4550 microcontroller were previously used in the Microcontrollers course and continue to be used in the Embedded Systems course, thus avoiding that students need to know the software and the structure related to a new microcontroller. The FreeRTOS is adopted in the course due to its gratuity and compatibility with PIC18F4550 microcontroller, in addition to offering all the functionality of a real time operating system [15].

The Embedded Systems course is organized into topics, summarized in the Table 1.

In each week of the presential course, theorical classes are between 50 and 90 minutes in classroom, in which the topics are presented by the professor using Datashow. Practical classes are 100 minutes in laboratory, in which the students gathered in groups use computer and development kit based on the PIC18F4550 microcontroller to implement the exercises.

At the 8th week, the students send the first list of exercises implemented in laboratory on institutional Moodle and solve the first writing test. At the 15th week, the students send the second list of exercises on institutional Moodle implemented in laboratory and solve the second writing test. If necessary, the students solve a complementary writing test at the 17th week.

TABLE I
TOPICS DURATION OF THE EMBEDDED SYSTEMS COURSE.

Topics	Description	Duration
Firmware architectures	The cooperative multitasking is presented like an alternative to one single loop for that tasks can be performed at certain times. Examples applied in embedded systems are also presented	Weeks 1-3
RTOS	The functional characteristics of the real time operation system, with emphasis on preemption, are presented and compared to the functional characteristics of the cooperative multitasking	Week 4
FreeRTOS	The FreeRTOS functions regarding tasks management and examples applied in embedded systems are presented	Weeks 5-7
Queues	The functional structure of queues, FreeRTOS functions regarding queues management and examples applied in embedded systems are presented	Week 9
Binary semaphores	The functional structure of binary semaphores, FreeRTOS functions regarding binary semaphores management and examples applied in embedded systems are presented	Week 10
Counting semaphores	The functional structure of counting semaphores, FreeRTOS functions regarding counting semaphores management and examples applied in embedded systems are presented	Week 11
Event group	The functional structure of event group, FreeRTOS functions regarding event group management and examples applied in embedded systems are presented	Week 12
Resources management	Resource management using critical sections and binary semaphores are explored in this topic through examples applied in embedded systems	Weeks 13-14

II. EMBEDDED SYSTEMS REMOTE COURSE

In the embedded systems remote course are presented the same topics showed in the Table 1. In each week, on-line theorical classes are 50 minutes using the Google Meet. The presence of students is not mandatory and classes are recorded and made available on the institutional Moodle for later viewing. In these classes the content of each topic is explored through practical examples based in C codes using the MPLAB and C18 compiler (shown in Figure 1). A document with the content of each topic and the practical examples are previously available to students on institutional Moodle.

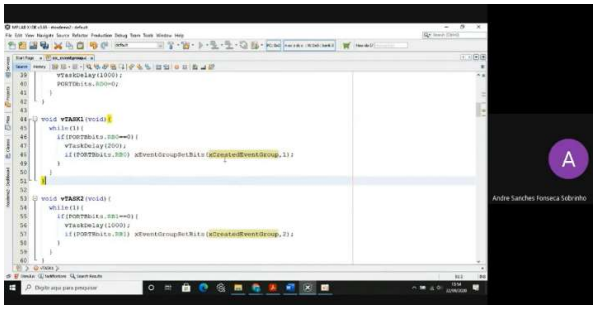

Figure 1. On-line theorical class using Google Meet, Mplab and C18 compiler.

In the same on-line theorical classes, the compiled C codes are subsequently implemented in the PicSimLab free software. PICSimLab means PIC Simulator Laboratory and it is a real-time emulator of development boards with integrated MPLAB debugger. It supports some PIC microcontrollers and has functionality almost identical to the development kit used by the students in laboratory classes (shown in Figure 2), allowing some hardware manipulation that cannot be reached using virtual laboratories [16, 17].

Figure 2. Development kit used by the students in laboratory classes.

Figure 3 shows the screen of an on-line theorical class using PicSimLab software. Practical classes are replaced by offline activities, in which the students gathered in remote groups use their own computers and the PicSimLab to implement the exercises of the lists to be delivered later on institutional Moodle. The writing tests are replaced by exercises lists personalized for each remote student group that must be delivered within 1 day on institutional Moodle.

Figure 3. On-line theorical class using PicSimLab software.

The topics of the embedded systems remote course are briefly explored below, using exercises showed in on-line theorical classes and included in the student's exercises lists.

A. Firmware architectures

This topic covers the use of cooperative multitasking architecture as an alternative to one single loop, which doesn't ensure that tasks are executed within defined time interval [18]. The cooperative multitasking architecture, besides to use a microcontroller timer to ensure that all tasks are executed within a defined time interval, it also to

run few tasks in each loop ensuring that the sum of time duration of the tasks is less than the defined time interval.

Exercise 1: Implement a cooperative multitasking system based on the PIC18F4550 [19] using the PicSimLab software, in which the following tasks must be performed without delay:

- a first square wave must be generated with 6.25Hz frequency and 50% duty cycle on the I/O pin RD1, that is, 80ms at high logic level and 80ms at low logic level;

- a second square wave must be generated with 3.125Hz frequency and 50% duty cycle on the I/O pin RD2, that is, 160ms at high logic level and 160ms at low logic level;

- a third square wave must be generated with 1.5625Hz frequency and 50% duty cycle on the I/O pin RD3, that is, 320ms at high logic level and 320ms at low logic level.

Note 1: maximum two tasks in each loop.

Note 2: show the code, the scope screen with RD1 and RD2 pins and the scope screen with RD3 pin.

In the code a microcontroller timer must be configured with 80ms base time, once this value being multiple for all tasks. The first task must be allocated in the *top slot* of the cooperative multitasking algorithm to be run in each 80ms. The others tasks must be allocated in the *cases* of the cooperative multitask algorithm to be run in each 160ms. The third task will need an additional logic to be run in each 320ms. Maximum two tasks (Task1/Task2 or Task1/Task3) are run in each loop (shown in Figure 4. Tasks behavior are verified using the "Oscilloscope" option in PicSimLab, which the pins, V/div scale and ms/div scale can be configured (shown in Figure 4).

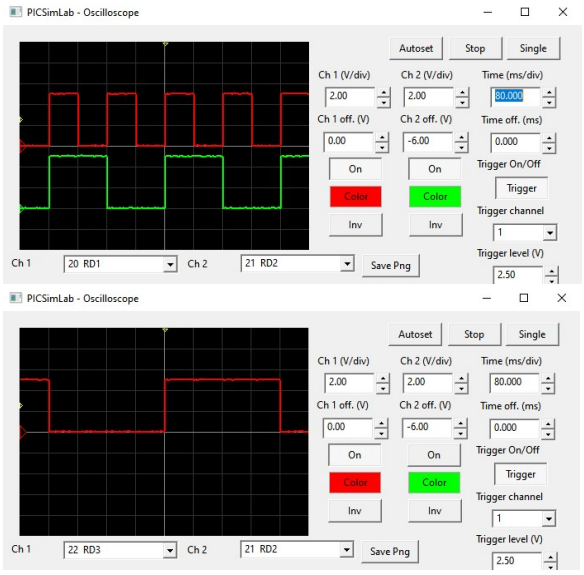

Figure 4. Scope screen with RD1 and RD2 and scope screen with RD3 for Exercise 1, with 2V/div scale and 80ms/div scale.

B. RTOS and FreeRTOS

The functional characteristics of the real time operation system, with emphasis on preemption, task scheduling and task priority level are presented in this topic and compared to the functional characteristics of the cooperative multitasking [20]. The FreeRTOS is adopted in the course, as previously mentioned in Section 2, and functions regarding creation and management of tasks are explored.

Exercise 2: Implement the Exercise 1 using the FreeRTOS.

Note: show the code, the scope screen with RD1 and RD2 pins and the scope screen with RD3 pin.

With FreeRTOS will be necessary to use just three independent tasks. The advantage of using an RTOS is even more evident to students when implementing Exercise 3:

Exercise 3: add to Exercise 2 a fourth task in which a square wave must be generated with 10Hz frequency and 60% duty cycle on the RD4 pin (60ms at high logic level and 40ms at low logic level).

Note1: An RTOS may or may not be used, but the tasks must be performed without delay;

Note2: show the code, the scope screen with RD1 and RD2 pins and the scope screen with RD3 and RD4 pins.

Students should verify that the execution time in which this new task is repeated is not a multiple of 80ms, and without the use of FreeRTOS it will be necessary to create a new state machine, much more complex, with a great chance of failures and that will affect tasks that were already implemented. With the use of FreeRTOS, a fourth task will be created independently (shown in Figure 5), without impacting the other tasks already implemented.

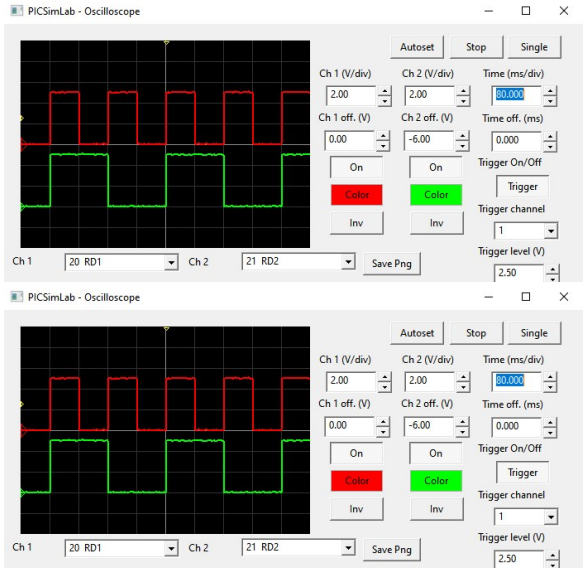

Figure 5. Scope screen with RD1 and RD2 and scope screen with RD3 and RD4 for Exercise 3.

C. Queues

In this topic are presented the functional structure of queues, mainly regarding the writing and reading of sequential dada, using the FreeRTOS functions [15].

Exercise 4: Implement a system based on the PIC18F4550 using the PicSimLab software and the FreeRTOS, in which the following tasks must be performed:

- check if the RB0, RB1 and RB2 button were pressed and then released with 100ms delay;

- if the RB0 button was pressed and then released, D0 LED should flash with 2.5Hz frequency and 50% duty-cycle for 2s. For RB1 button, D0 LED should flash with 5Hz frequency and 50% duty-cycle for 2s. For RB2 button, D0 LED should flash with 10Hz frequency and 50% duty-cycle for 2s.

Note 1: it is important that regardless of the number of times the buttons are pressed and released, D0 LED is activated the same number of times and in the order in which the buttons were pressed;

Note 2: button pressed is equivalent to logic level 0;

Note 3: D0 LED is interfaced with RD0 pin;

Note 4: show the code and the scope screens for each D0 LED frequency.

The use of the queue ensures that D0 LED is activated following the number of times and in the order in which the buttons were pressed (Figure 6). It's also important highlight in the code that the task that waits data in the queue keeps in the blocked state while the queue is empty.

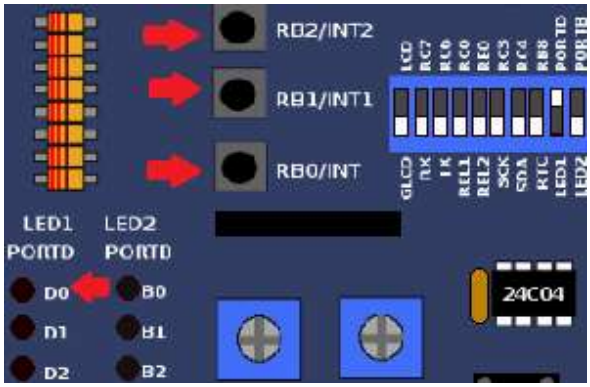

Figure 6. Buttons and D0 LED used in Exercise 4.

D. Binary Semaphores

In this topic are presented the functional structure of binary semaphores, mainly regarding the activation and verification of the semaphore, using the FreeRTOS functions [15].

Exercise 5: Implement a system based on the PIC18F4550 using the PicSimLab software and the FreeRTOS, in which the following tasks must be performed:

- check the receipt through serial communication (9600 bps) of the following messages: "D1","D2", "D3","D4","D5","D6","D7","D8","D9","T2","T4","T6" and 'S'. After receiving any of these messages, the microcontroller should return "OK";

- the duty cycle of the 1Hz PWM signal to be generated on D1 LED must be configured with the messages between 'D1' and 'D9'. Ex: 'D2' -> 20%;

- the duration of the 1Hz PWM signal to be generated on D1 LED must be configured with the messages "T2" (2 seconds), "T4" (4 seconds) and "T6" (6 seconds);

- the 1Hz PWM signal must be generated only with the receipt of the 'S' character.

- toggle D0 LED each 100ms;

Note: show the code, a scope screen with D0 and D1 LEDs for the 70% duty cycle setting and 6s duration (show the serial terminal screen (CuteCom) with the answer to these settings).

The serial communication can be verified using the "Serial Terminal" option in PicSimLab (shown in Figure 7).

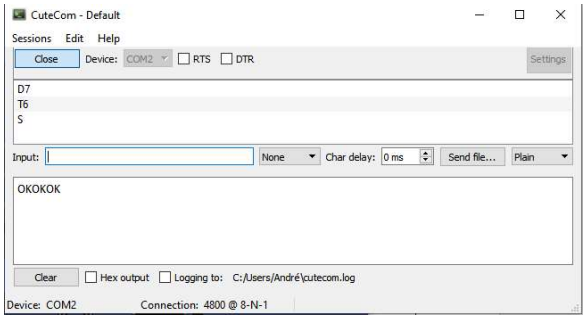

Figure 7. Serial Terminal for Exercise 5.

Students should verify that the detection of the characters must occur as soon as possible by the microcontroller (fast handler) in the interruption, activating a semaphore that will unblock the task responsible for response message and the generation of the PWM signal (slow handler).

E. Counting semaphores

The functional structure of counting semaphores is presented in this topic as an evolution of the binary semaphores, which is "memorized" the number of times that the semaphore is activated. FreeRTOS functions [15] regarding counting semaphores management also are presented.

Exercise 6: Implement a system based on the PIC18F4550 using the PicSimLab software (shown in Figure 8) and the FreeRTOS, in which the following tasks must be performed:

- check if the RB0 button was pressed and then released with 100ms delay;

- check if the RB1 button was pressed and then released with 100ms delay;

- if the RB0 button was pressed and then released, D0 LED should flash with 10Hz frequency and 50% duty-cycle for 2s.

- if the RB1 button was pressed and then released, the cooler works for 2s.

Note 1: it is important that regardless of the number of times the buttons are pressed and released, D0 LED and cooler are activated the same number of times in which the buttons were pressed;

Note 2: it is important that the execution of a task that "depends" on another task is in the blocked state whenever possible;

Note 3: button pressed is equivalent to logic level 0;

Note 4: D0 LED is interfaced with RD0 pin and the cooler is interfaced with pin RC2 pin.

Note 5: show the code, a scope screen with D0 LED and a scope screen with RC2 pin.

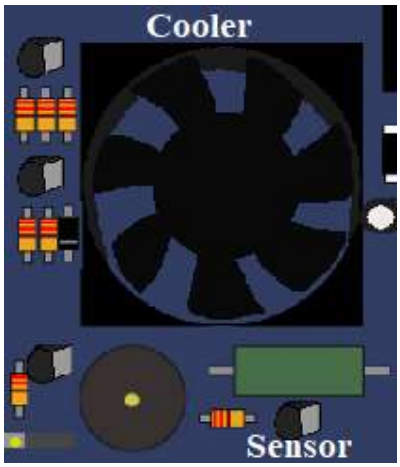

Figure 8. Cooler used in Exercises 6 and 8.

The use of the counting semaphores ensures that D0 LED and the cooler are activated following the number of times in which the buttons were pressed. Unlike the queue, it is not necessary to distinguish different tasks that block a same task.

It's also important highlight in the code that the task that waits the activated semaphores keeps in the blocked state while the semaphores is disable.

F. Event group

In this topic are presented the functional structure of event group, mainly regarding the activation, distinction of events and verification of the activated events, using the FreeRTOS functions [15].

Exercise 7: Implement again the Exercise 4. It's not important that regardless of the number of times the buttons are pressed and released, D0 LED is activated the same number of times and in the order in which the buttons were pressed.

Note 1: it is important that the execution of a task that "depends" on another task is in the blocked state whenever possible;

Note 2: show the code and the scope screens for each D0 LED frequency.

Students should verify that event group using unlike the queue is possible because It isn't important that D0 LED is activated the same number of times and in the order in which the buttons were pressed. The required distinction of events for this exercise is correctly provided by event group and requires less program and data memory than using the queue.

G. Resources management

Resource management using critical sections and binary semaphores are explored in this topic, using the FreeRTOS functions [15].

Exercise 8: Implement a system based on the PIC18F4550 using the PicSimLab software and the FreeRTOS (shown in Figure 8), in which the following tasks must be performed:

- PicSimLab has a temperature sensor with a linear behavior of 2.5mV/ºC. Make a temperature control, which keeps the temperature read by the sensor between 30 and 40ºC. Read the temperature of the sensor using the A/D converter every 3s and with the heating resistance always on, turn on the fan when the temperature is above 40ºC and turn off it when the temperature is below 30ºC. Send the temperature value to the PC via serial communication.

- Every time the "T" character is received from the PC, the temperature value must be sent immediately to the PC;

- Every time the "L" character is received from the PC, the fan must be turn on for 4 seconds continuously, without interruptions;

- Every time the "D" character is received from the PC, the fan must be turn off for 4 seconds continuously, without interruptions.

Note 1: temperature sensor is interfaced with RA2 pin, heating resistor is interfaced with RC5 pin and the cooler is interfaced with RC2 pin;

Note 2: show the code and the serial terminal screen (CuteCom) with the answer to the messages.

In this exercise, the students should adopt the critical section in the task where the cooler must be turned on or turned off for 4 seconds continuously, without interruptions of the task that every 3s could change the cooler operation.

III. RESULTS

The Embedded System course is a compulsory semester course for undergraduates of the Control and Automation Engineering, Electronic Engineering and Computer Engineering.

By the end of the Embedded Systems remote course, a questionnaire was applied to the students using Google Forms. In addition to the 5 questions, the questionnaire allowed students to comment about the course and contribute with suggestions. Tables 2-6 show the percentages of the answers for each question, followed by comments from the author and the students.

Most students agree that PicSimLab successfully replaced the use of the development kit for practical learning. A common comment of the students is that although the software does not allow to see in practice the functioning of the system, as the development kit, the software was essential for the remote course.

TABLE II
QUESTION 1

The development kit based on the PIC18F4550 was used in the lab classes of Microcontrollers course and at the beginning of Embedded System. Has the use of PicSimLab successfully replaced the use of the development kit for practical learning in the Embedded Systems remote course?	**Electronic engineering students**	**Computer engineering students**	**Control and automation engineering students**
Response A: I totally agree	66.7%	88.9%	85.7%
Response B: I agree in part	33.3%	11.1%	14.3%
Response C: I don't agree	0%	0%	0%

TABLE III
QUESTION 2

Even with the return of presential classes, should the PicsimLab software be adopted at least as a support tool for practical learning in the Embedded Systems course?	**Electronic engineering students**	**Computer engineering students**	**Control and automation engineering students**
Response A: I totally agree	83.3%	100%	85.7%
Response B: I agree in part	0%	0%	14.3%
Response C: I don't agree	16.7%	0%	0%

For this question, the most of students also agreed that the PicSimLab should be adopted in some way in the Embedded Systems in-class course. The sstudent's commented that using the software it is always possible to test the codes any time, which helps in learning.

TABLE IV
QUESTION 3

How do you evaluate the other tools (Google Meet, Moodle and other software) used in the remote course?	**Electronic engineering students**	**Computer engineering students**	**Control and automation engineering students**
Response A: Excellent	58.3%	55.6%	85.7%
Response B: Very good	41.7%	44.4%	14.3%
Response C: Good	0%	0%	0%
Response D: Regular	0%	0%	0%
Response E: Bad	0%	0%	0%

All the students agree that the set of software and on-line services used in the Embedded Systems remote course was excellent or very good. In the comments, the students highlighted that the possibility of watching the recorded classes again through google meet significantly contributed to the understanding of the course, allowing later doubts to be resolved.

TABLE V
QUESTION 4

How do you rate the evaluation system employed in the Embedded Systems remote course?	**Electronic engineering students**	**Computer engineering students**	**Control and automation engineering students**
Response A: Excellent	58.4%	55.6%	78.6.3%
Response B: Very good	33.3%	44.4%	21.3%
Response C: Good	8.3%	0%	7.1%
Response D: Regular	0%	0%	0%
Response E: Bad	0%	0%	0%

For this question, not all the students rated the evaluation system employed in the remote course as excellent or very good, which indicates a need for improvement. A common suggestion is for students to develop a personal project with the system functionality subsequently demonstrated and evaluated through a report.

TABLE VI
QUESTION 5

Overall, how do you rate the Embedded Systems remote course?	**Electronic engineering students**	**Computer engineering students**	**Control and automation engineering students**
Response A: Excellent	58.3%	66.7%	78.6%
Response B: Very good	41.7%	33.3%	21.4%
Response C: Good	0%	0%	0%
Response D: Regular	0%	0%	0%
Response E: Bad	0%	0%	0%

All the students rated the course as excellent or very good, even with the necessary adaptations to offer the course in remote mode.

IV. CONCLUSION

This paper presents an embedded systems remote course that was taught in different engineering bachelors at Universidade Tecnológica Federal do Paraná (UTFPR), Cornélio Procópio campus.

Even with the excellent or very good evaluation of the course by the students, mainly regarding the use of the

PicSimLab to explore the topics of the course in the on-line theorical classes and to implement the exercises of the lists, a necessary improvement in the course is the inclusion of a personal project as an evaluation item. However, contrary to suggestion of the students regarding the evaluation of the project through a report, a remote evaluation would also be necessary, in which the functionality system could be demonstrated and the students could be questioned.

The positive evaluation of students encourages the continuous improvement of the embedded systems remote course and its application even in a scenario that allows the realization of presential classes.

REFERENCES

[1] H. Lim, H. Yu and T. Suh, "Using Virtual Platform in Embedded System Education," *Comput Appl Eng Educ*, vol. 20, pp. 346–355, 2012.

[2] M. Jiménez, R. Palomera and I. Couvertier, *Introduction to Embedded Systems*. New York: Springer, 2014.

[3] D. M. Laverty, J. Milliken, M. Milford and M. Cregan, "Embedded C programming: a practical course introducing programmable microprocessors," *European Journal of Engineering Education*, vol. 37, 2012.

[4] Y. Luo, S. Qin and D. Wang, "Reform of embedded system experimente course based on engineering education accreditation," *International Journal of Electrical Engineering & Education,* pp. 1–14, 2020.

[5] Texas A. M. University, *Computer Engineering 2020-2021 Full Degree Plan*. https://engineering.tamu.edu/cse/academics/degrees/undergraduate/bs-ce.html

[6] Korea University, *Official Study Plan for Bachelor in Electrical Engineering.* http://eng.korea.edu/ee_en/subject/ course_guide.do

[7] University of Cape Town, *Official Study Plan for Bachelor of Science in Engineering in Electrical and Computer Engineering*. http://www.ee.uct.ac.za/bachelor-science-engineering-electrical-and-computer-engineering

[8] Victoria University, *Official Study Plan for Electrical and Electronic Engineering BEng.* https://www.vu.edu.au/courses/bachelor-of-engineering-honours-electrical-and-electronic-engineering-nhee

[9] Coventry University*, Official Study Plan for Electrical and Electronic Engineering MEng/BEng.* https://www.coventry.ac.uk/course-structure/ug/2020-21/eec/electrical-and-electronic-engineering-beng

[10] Universidade do Porto, *Official Study Plan for the Integrated Master in Electrical and Computer Engineering*. https://sigarra.up.pt/feup/pt/ucurr_geral.ficha_uc_view?pv_ocorrencia_id=436930

[11] Microchip Inc., *C18 Compiler version 3.46*, 2019.

[12] Microchip Inc., *Mplab IDE version 8.2*, 2009.

[13] R. T. Engineers, *What is a RTOS? [Online]*. Available at http://www.freertos.org/about-RTOS.html.

[14] L. C. Gamboa, *PicSimLab version 0.8.1 [Online]*. Available at https://github.com/lcgamboa/picsimlab/releases.

[15] R. Barry*, Mastering the FreeRTOS Real Time Kernel.* Bristol: Real Time Engineers Ltd., 2016.

[16] X. Vilajosana, J. Llosa, I. Vilajosana and J. Prieto-Blàsquez, "Arp@: Remote Experiences with Real Embedded Systems," *Comput Appl Eng Educ*, vol. 22, pp. 639-648, 2014.

[17] S. Chtourou, M. Kharrat, N. B. Amor, M. Jallouli and M. Abid," Using IOIOAI in introductory courses to embedded systems for engineering students: a case study," *International Journal of Electrical Engineering & Education*, vol. 55, pp. 62–78, 2018.

[18] Kopják and J. Kovács, 2012. Timed cooperative multitask for tiny real-time embedded system. In *IEEE 10th International Symposium on Applied Machine Intelligence and Informatics*.

[19] Microchip Inc, *8-bit PIC Microcontroller Peripheral Integration Quick Reference Guide,* 2018.

[20] M. Siegesmund, *Embedded C Programming: Techniques and Applications of C and PIC MCUS*. Boston: Newnes, 2015.

AUTHORS

André Sanches Fonseca Sobrinho is an Assistant Professor at Electrical Engineering Department, Universidade Tecnológica Federal do Paraná, Cornélio Procópio, PR Brazil, (e-mail: andresobrinho@ utfpr.edu.br).

This work was supported in part by Universidade Tecnológica Federal do Paraná.