Enhancing Teaching Methods on Embedded Systems with Project-Based Learning

Filippo Sanfilippo
Dept. of Engineering Cybernetics
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology (NTNU)
7491 Trondheim, Norway
Dept. of Science and Industry Systems
Faculty of Technology, Natural Sciences and Maritime Sciences,
University of Southeast Norway (USN)
Post box 235, 3603 Kongsberg, Norway
filippo.sanfilippo@usn.no

Kolbjørn Austreng
Dept. of Engineering Cybernetics
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology (NTNU)
7491 Trondheim, Norway

***Abstract*—Automation engineering departments must continuously develop their laboratories and pedagogical tools to provide their students with effective study plans. While acquiring state-of-the-art equipment can be financially demanding, an effort is made at the Norwegian University of Science and Technology (NTNU) in Trondheim to provide the students with a hands-on sustainable experience. A strategy that consists of adopting low-cost commercial off-the-shelf (COTS) components for learning purposes is selected. This combines both industry-standard automation controllers, such as Programmable Logic Controller (PLC) technology, as well as novel microcontrollers designed for use in embedded systems education. Specifically, the micro:bit microcontroller based on the nRF51822 system-on-chip (SoC) and designed by the British Broadcasting Corporation (BBC) is adopted. This choice is supported by an agreement between NTNU and the Norwegian company Nordic Semiconductor, which produces the nRF51822 SoC. This paper proposes a novel organisation of the embedded systems module for the engineering cybernetics education curriculum. Students are engaged in both a series of theoretical lectures as well as practical and highly-involving laboratory group projects. The course organisation and main topics as well as result analysis of student surveys are discussed. The survey results indicate that the course organisation and topics are effective for the students.**

Keywords—embedded systems; education; programming; micro:bit

I. Introduction

In this paper, an innovative organisation of the embedded systems module for the engineering cybernetics education curriculum is proposed. As shown in Figure 1, the underlying idea is to organise the course into three parallel layers:

- theoretical lectures with exercises. This component of the module is systems-oriented and focuses on system specifications, modelling, development processes, performance estimation, verification, architecture design and control;
- laboratory. This component of the course is designed to be both hardware-oriented, by focusing on studying the industry standard Programmable Logic Controller (PLC) [1] and the advanced reduced instruction set computing (RISC) machine (ARM)-based embedded system micro:bit [2] designed by the British Broadcasting Corporation (BBC), as well as software-oriented, by focusing on programming, debugging and reviewing. The laboratory is based on group projects;
- applications. Both the theoretical lectures as well as the laboratory work is designed to provide the students with an improved hands-on automation experience for implementing industry standard embedded systems [3].

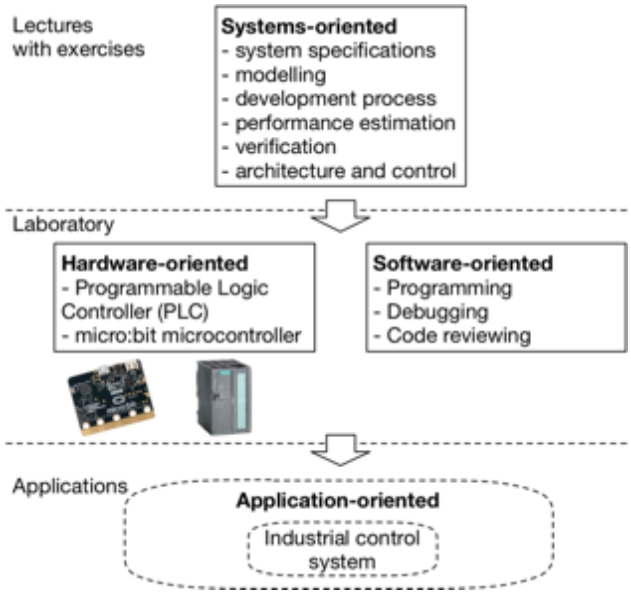

Fig. 1. The proposed hands-on organisation of the embedded systems course.

The presented course is the TTK4235 - Embedded Systems [4]. This course is a 4th semester module of the five years master's degree programme in cybernetics and robotics given at the Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway. Recommended previous knowledge for the course includes basic knowledge and skills in the fields of analog and digital electronics, information technology and programming. Before taking this module, all the students are required to attend extensive courses on both procedural and object-oriented programming. The course organisation and main topics are

presented in this paper. To show the effectiveness of the course from a pedagogical perspective, result analysis of after-course surveys are outlined. The comparison between the results of the surveys indicates that the module design and topics are engaging, effective and helpful for students. The paper is organised as follows. In Sect. II, the selected pedagogical tools are presented. The course overview is discussed in Sect. III. The laboratory content is depicted in Sect. IV. The course learning outcomes are delineated and analysed in Sect. V. Conclusions and future work are outlined in Sect. VI.

II. Pedagogical Tools

The adoption of the unified modelling language (UML) [5] as a foundation for the proposed module is motivated by the fact that this tool can potentially be utilised to build a solid educational and scientific base with embedded systems design as the cornerstone, which will ensure a systematic and even-handed integration of concepts from both computer science and electrical engineering [6].

The choice of the BBC micro:bit platform [2] is motivated by a variety of reasons. Firstly, it is desirable to use an ARM-based architecture, as NTNU already has courses focusing on other architectures [7]. This requirement is met by the Nordic Semiconductor nRF51822 System on Chip [8], which is the main component of the BBC micro:bit platform. Furthermore, for practical and logistical reasons, it is desirable to use a "self-contained" board that would limit the need for the students to do extensive manual setup before the lab work could begin. The "plug-and-play" aspect of the BBC micro:bit proved excellent for this. Lastly, the assumption that students learn best by tinkering with systems on their own, made it desirable that each student would get to keep their lab equipment after the lab sessions – and after the course all together. The micro:bit is an inexpensive piece of equipment; as well as being easy to extend outside of a lab setting. These characteristics make the micro:bit the ideal platform for teaching embedded systems in a practical way. It should be noted that there are many programming languages available for the micro:bit, including Python, Touch Develop, Javascript and C++ [2]. However, the C language has been selected for the proposed course as the main programming language to be used. This choice forces the students to code their applications without relying on extra support from existing software libraries. This fact exposes them to a much deeper understating of what is happening at a lower level from a software/hardware perspective, highlighting important aspects and challenges that are typical of embedded systems.

The selection of PLC-based developing platforms for the proposed course is motivated by the fact that this is an effective way of providing students with a real industry-like experience. In [9], a cost-effective approach for the design of educational projects in a PLC course for electrical engineering education is presented. In [10], a PLC platform is used to recycle a discarded robotic arm for automation engineering education. These works show that engineering students improve their practical problem-solving abilities by working on an extensive design project using PLC-based technology.

TABLE I. The Organisation of The Course Content

Lectures	Laboratory	Projects
8 lectures	hands-on laboratory classes	elevator project (PLC)
		elevator project (C-programming)
		micro:bit (C-programming) project

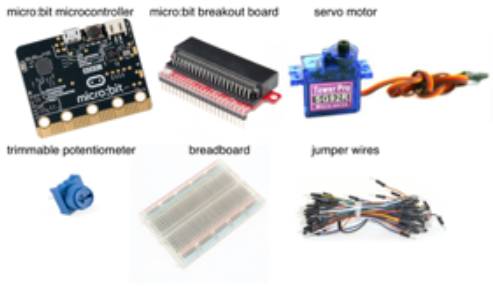

Fig. 2. Each student is provided with a micro:bit starter kit consisting of a micro:bit microcontroller, a micro:bit breakout board, a servo motor, a trimmable potentiometer, a breadboard and set of jumper wires.

III. Course Overview

In this section, the proposed course organisation and main topics are presented. As shown in Table I, the course content includes 8 theoretical lectures, laboratory classes and 3 course projects. The content of the theoretical lectures is depicted in Table II. Each weekly lecture lasts 6 hours. The topics of each lecture are presented hereafter.

Lecture 1: introduction on embedded systems and UML, Nordic Semiconductor seminar. Lecture 1 presents the course overview, expectations, logistics, processes, syllabus and a session of questions and answers related to the course and to the corresponding prerequisite material [4]. Successively, an introduction of embedded system with the most relevant descriptions, definitions and vocabulary is considered [11]. Following, a review of microprocessor/microcontroller architectures is discussed. To engage the students and introduce them to the forthcoming laboratory work, a short seminar is organised by the company Nordic Semiconductor. During this seminar, each student receives a micro:bit starter kit consisting of a micro:bit microcontroller, a micro:bit breakout board, a servo motor, a trimmable potentiometer that has a small knob built right in and it is breadboard friendly, a breadboard and set of jumper wires. This starter kit is shown in Figure 2. According to a formal agreement between Nordic Semiconductor and NTNU, the company provides the microcontrollers while all the other components are provided by the university. This arrangement contributes towards a hands-on sustainable learning experience and it enables students to use low-cost, course-specific hardware to complete lab exercises at home. This represents an extension of the university laboratory and gives students the possibility of improving their learning involvement. Lecture 1 is additionally complemented with an introduction of UML. An overview of different paradigms and models regarding the development process of embedded systems is also given with particular emphasis on designing techniques, such as incremental and waterfall approaches [12]. Finally, an introduction of UML class diagrams is given [5].

Lecture 2: flipped class on C-programming, PLC introduction. Lecture 2 consists of a review of the main topics of the C programming language [13]. Motivated by the fact that the presented course is a 4th semester module of the five years master's degree programme in cybernetics and robotics, and that all the students have already attended extensive courses on both procedural and object-oriented programming, this class is organised as a flipped classroom [14] covering the following topics: programming introduction, overview of how the C language works, types, operators and expressions, control flow, functions and program structure, pointers and arrays, structures, input and output. A flipped classroom is an instructional strategy and a type of blended learning that reverses the traditional learning environment. This is achieved within this course by delivering instructional content and describing the main guidelines for each considered topic of the lecture beforehand through the on-line Blackboard [15], a virtual learning environment and course management system. In this way, the students are divided in groups and can prepare their assigned topics so that the class becomes the place to work through problems, advance concepts, and engage in collaborative learning. Lecture 2 is additionally complemented with an introduction to Programmable Logic Controller (PLC) [1], [16] programming to introduce the students to the forthcoming laboratory work. Finally, a discussion on C-programming and coding style is given to enable student making the best use of the C language [17].

Lecture 3: UML concepts and class exercise. Lecture 3 introduces the fundamental concepts of UML sequence diagrams [5], [18], [19] with particular emphasis on embedded systems. These diagrams allow for getting clear visual clues to possible flows of control over time, for emphasising time ordering, for showing object lifelines and for illustrating the focus of control. The notion of messages (or stimulus) and of lifeline are discussed by highlighting the observation of time, temporal constraints and object activations. The concepts of suspension, interaction, duration constraints are also outlined. Successively, the UML use case diagrams are presented as an essential tool for identifying services offered by the system to be designed and its main functionalities. The different concepts of inclusion, extension and generalisation are analysed. Further, advanced concepts related to UML class diagrams are outlined, such as keywords, multiple and dynamic classification, associations, enumerations, responsibilities, static operations and attributes, aggregation and composition, derived properties and qualified associations. To introduce the students to the forthcoming laboratory work, a class exercise is finally considered focusing on the development of use cases, class and sequence diagrams for an elevator system.

Lecture 4: UML concepts and class exercise. Lecture 4 depicts the essential concepts of UML state machine diagrams [5]. A finite state machine is a popular technique to describe the behaviour of a system and it is also one of the most relevant design patterns in embedded systems. Many applications from simple home appliances to complex communication systems implement event-based state machines. Different aspects are discussed within this lecture, including internal activities, activity states and super states. With particular emphasis on embedded systems, the design of concurrent states is discussed allowing the students to anticipate both the benefits and challenges of concurrent programming. Guidelines for implementing state machines with the C-programming language are successively outlined by highlighting the use of nested switches to handle the state transitions. Successively, UML timing diagrams are described as another form of interaction diagrams, where the focus is on timing constraints. When considering embedded systems, UML timing diagrams are extremely relevant to identify time constraints and deadlines. Finally, a class exercise is considered to prepare the students to the forthcoming laboratory work by focusing on the development of UML state machine and timing diagrams for an elevator system.

Lecture 5: analog and digital signals, communication, code verification. Lecture 5 presents an introduction and review of analog and digital signals [20]–[22]. The difference between analog and digital signals is described outlining their most important properties. Definitions about certain elementary signals are provided. The basic notions involved in the characterisation of communication systems are outlined. With respect to embedded systems, it is highlighted that working with electronics means dealing with both analog and digital signals, inputs and outputs. Electronics systems have to interact with the real, analog world in some way, but most of our microprocessors, computers, and logic units are purely digital components. These two types of signals are like different electronic languages. Signals are passed between devices in order to send and receive information. To achieve this, different basic notions of communication protocols are presented. In particular, the two main properties of data exchange are discussed: message-based data exchange and shared memory-based data exchange. The definition of the most essential communication parameters is depicted, such as latency, jitter, fault handling and redundancy. Based on these fundamental concepts, it is highlighted in this lecture that embedded electronics is based on interlinking circuits (processors or other integrated circuits) that are integrated to create a symbiotic system [22]. Individual circuits must share a common communication protocol to swap their information. Hundreds of communication protocols exist, and, in general, each can be separated into one of two categories: parallel and serial. Based on this classification, the following communication protocols are presented in the class: serial, universal asynchronous receiver transmitter (UART), serial peripheral interface (SPI), inter-integrated circuit (I2C) [21], [22]. To prepare the students to the forthcoming laboratory work, a review of code verification techniques for C-programming is finally presented [23].

Lecture 6: ADC/DAC, modulation, SDLC, class exercise. Lecture 6 introduces a review of the fundamental notions for converting a signal from analog (continuous) to digital (discrete) form [21]. This conversion, which is achieved by adopting analog-to-digital converters (ADC), is especially relevant for embedded systems because it provides a link between the analog world of transducers/sensors and the digital world of signal processing and data handling. The main steps of the conversion process are described in detail, including the phases of sampling and holding, as well as quantisation and encoding. Relevant sampling considerations are discussed by

TABLE II. THE ORGANISATION OF THE COURSE TEORETICAL LECTURES

Week	Course content	Description	References	Time
Week 1	Lecture 1	- Course overview, expectations, logistics, processes, syllabus, FAQ, and prerequisite material	[4]	2 hours
		- Embedded systems descriptions, definitions and vocabulary - Microprocessor/microcontroller architectures - micro:bit introduction: Nordic Semiconductor Seminar	[2], [11], [31]	2 hours
		- UML introduction; paradigms and models (development process): prototyping, incremental, waterfall - UML: Class Diagrams	[5], [12]	2 hours
Week 3	Lecture 2	- Flipped class on C-programming review: introduction, how C works, types, operators and expressions, control flow, functions and program structure, pointers and arrays, structures, input and output	[13]	5 hours
		- PLC introduction: "Programming of a FESTO production line"	[1], [16]	0.5 hours
		- C-programming: coding style, making the best use of C	[17]	0.5 hours
Week 5	Lecture 3	- UML Sequence Diagrams, Use cases, Class Diagrams, advanced concepts	[5], [18], [19]	4 hours
		- Class exercise: Use cases, Class Diagram and Sequence Diagrams for the Elevator project		2 hours
Week 7	Lecture 4	- UML: State Machines and Timing Diagrams	[5]	4 hours
		- Class exercises: State Machines and Timing Diagrams for the Elevator project		2 hours
Week 9	Lecture 5	- Analog and digital signals	[20], [21]	1 hour
		- Communication protocols: serial, UART, SPI, I2C	[22], [31]	4 hours
		- C-programming: code verification	[23]	1 hour
Week 11	Lecture 6	- ADC and DAC, aliasing	[21]	1 hour
		- Modulation (AM, FM, PWM)	[21]	2 hours
		- Software Development Cycle: non-functional Vs functional requirements, Waterfall Vs Iterative V Model	[5]	1 hours
		- Class exercises: UML modelling of an automated teller machine (ATM)		2 hours
Week 14	Lecture 7	- Industrial instrumentation and control	[29]	2 hours
		- Temperature control		1 hour
		- Error in measurements		1 hour
		- Class exercises: UML modelling of a Vending Machine		2 hours
Week 16	Lecture 8	- Reading research papers	[30]	2 hours
		- Exam simulation: requirements analysis, development process selection, system design, sensors		4 hours

highlighting the importance of the sampling frequency in terms of reconstructing the transmitted signal. In this perspective, the sampling theorem is outlined as a fundamental bridge between continuous-time signals and discrete-time signals [21]. As a direct consequence of this theorem, the phenomenon of aliasing is described as an effect that causes different signals to become indistinguishable (or aliases of one another) when sampled with an improper frequency. The definitions of workspace, scope, dynamic range and resolutions are successively introduced and supported with various class exercises. Further, the necessity of designing communication strategies for interconnecting remote embedded systems is discussed by highlighting that the objective of a communication system is to transmit information signals (baseband signals) through a communication channel [21]. Since this baseband signal must be transmitted through a communication channel, an appropriate procedure is required to shift the range of baseband frequencies to other frequency ranges suitable for transmission, and a corresponding shift back to the original frequency range after reception. This is known as the process of modulation and demodulation. Based on these concepts, a review of different modulation techniques is presented, including amplitude modulation (AM), frequency modulation (FM) and pulse width modulation (PWM). To support the students with the forthcoming laboratory work and projects that are run in parallel, a discussion on the software development cycle is presented by highlighting the differences between non-functional and functional requirements and by introducing the V-model [24] software development life cycle (SDLC) as an extension of the previously introduced waterfall model. Finally, a class exercise is considered by focusing on the UML modelling of an automated teller machine (ATM).

Lecture 7: industrial instrumentation and control, errors, class exercise. Lecture 7 presents a review on control theory. Embedded systems traditionally follow the paradigm sense-think-act [25]. This require the use of sensors to monitor the environment, a decision-making approach to take a decision based on a predefined task and the sensed environment, and finally an acting mechanism, to perform the predefined task by adapting to the environment. Focusing on the thinking/decision-making necessity of embedded systems, fundamental notions of control theory are presented [26]. It is highlighted the fact that the majority of embedded designs are closed loop control systems, as opposed to open loop control. These concepts are especially relevant for students of engineering cybernetics. From this perspective, a review of essential notions for designing controllers for embedded systems is presented, including design guidelines for implementing a proportional–integral–derivative (PID) controller [27] and a bang–bang controller (2 step or on–off controller) [28], with practical applications to temperature

control for smart-buildings. Emphasising the fact that control and instrumentation are interdisciplinary fields, the basic concepts and principles that govern the operation of industrial plants and processes are successively discussed. In particular, the need for accurate measuring/sensing devices to achieve robust control of embedded systems is outlined. In this regard, the different types of error in measurements [29] is discussed from a qualitative point of view. Finally, a class exercise is considered by focusing on the UML modelling of an automated vending machine.

Lecture 8: reading research papers, exam simulation. Lecture 8 introduces some useful guidelines for reading research papers related to embedded systems. This is motivated by the fact that reading research articles is fundamental to stay up to date with the latest developments in embedded systems technology. This is additionally supported by the fact that more and more researchers recognise a mutual relationship between a student's academic reading skills and academic success [30]. Therefore, to provide the students with a hands-on reading and learning experience, the design and implementation of embedded systems is considered starting from their description through scientific papers. The design process includes the identification of the system requirement specifications, the selection of an appropriate development approach, the implementation of UML diagrams and the implementation of different aspects for sensors or communication protocols. Based on this methodology, a class simulation of the final exam is performed to prepare the students.

IV. Laboratory Overview

In this section, the proposed laboratory organisation and main topics are presented, as shown in Table III. The laboratory content is run in parallel with the theoretical lectures presented in Sect. III. In the following of this section, the main laboratory topics are presented referring to Table III.

PLC programming. The PLC laboratory of the lab is in many ways intended to prime the students with the necessary mindset of solving the more complex task of programming an elevator controller in C. The PLC implementation is not as "fully fledged" as the C implementation – but it is useful for understanding the needs of the system overall; something that student feedback has confirmed. The PLC lab setup consists of a Siemens SIMATIC S7-300 for each desk – connected to both a computer, as well as a model elevator, shown in Figure 3. The PLC is then programmed using the SIMATIC STEP-7 software [36]. As there are good manuals available for this [36], the students are left to discover the basics on their own – while being able to ask the on-lab student assistants for guidance where needed. The end goal of this exercise is to be able to control the model elevator to a specified floor by use of an accompanying order panel. This is a simplified version of the successive elevator C-programming project, in the sense that concurrent orders and a queue system is not expected from the students. The effectiveness of this lab module is demonstrated by the feedback collected from the students, as shown in Figure 4. After the PLC lab, students feel like they have a much better understanding of this topic.

Version control (Git). The main project of the course is

Fig. 3. The elevator model setup used in the course project. Courtesy of the Dept. of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

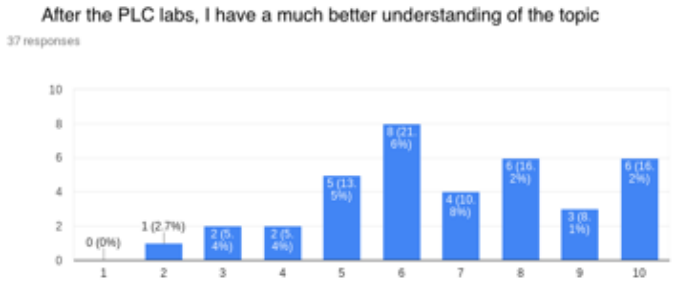

Fig. 4. Feedback collected from students regarding the helpfulness of conducting the PLC programming lab.

programming the elevator setup in the C programming language (see Sec. IV). As this design is much more complex than the PLC implementation, version control is necessary. Within this lab assignment, the students are introduced to the Git version control system [32]. The students are not forced to use Git over any other version control schemes, but it is expected that they can demonstrate that their code is under some version control. This is to encourage good, industry-proven approaches to manageable source code [37].

Debugging (GDB). With the increasing complexity of the systems to be developed, software bugs are a fact of life. When entering this course, the students are already familiar with a "printf-style" of debugging. In this course, they are encouraged to use more systematic, and less time-consuming, approaches [37]. To this end, they are introduced to the GNU Debugger (GDB) [33].

Elevator C Programming. The objective of the laboratory "Elevator C Project" is to program all the necessary control software to run the same elevator model as described in the PLC programming assignment. The model closely simulates an industry standard elevator and it consists of four floors, and an accompanying button panel for "cab orders" as well as "external orders". This setup is connected to a controlling computer, where the students write their software. The setup is shown in Figure 3. This laboratory project enables the students to get the most hands-on experience with programming a logical control system. At this stage, they already have an

understanding of the basic needs of the system, thanks to both the design methodology and the UML introduction from the theoretical lectures as well as the two-weeks PLC lab revolving around the same model. This understanding is further reinforced by having the students first document a suggested approach and the overall system architecture, before jumping head first into uncharted territories. When the students begin coding their solution, they quickly see the benefit of version control and debugging, which they have already been introduced to in weeks 4 and 5, as shown in Table III. Learning version control systems for the sake of version control and learning systematic debugging in a purely theoretical setting would make it difficult for the students to approach these concepts. However, being able to apply these concepts first-hand in a laboratory project where they are needed has been very beneficial for the retention of the material – as the class feedback has shown.

Elevator FAT and peer reviewing process. The elevator project culminates in a factory acceptance test (FAT), that is held 7 weeks after the beginning of the project, as shown in Table III. This acceptance test is scored and counts toward the

TABLE III. THE ORGANISATION OF THE LABORATORY COURSE WORK

Week	Lab content	Description	References
1	Group formation	Students form groups, either by choosing a lab partner, or by being assigned one.	
2	PLC programming	Introduction to PLCs in general. Implementing simple logic to get the first bit of hands-on experience.	
3	PLC programming	Students apply what they learned the previous week to implement a rudimentary elevator controller, that makes a model elevator go to a selected floor.	
4	Version control (Git)	Students are exposed to the most commonly used version control scheme today; git. At this stage, they learn what they will later need to know to effectively manage their own source code in the coming elevator project of the course.	[32]
5	Debugging (GDB)	Students already know how to do “print debugging” when taking this course. However, they lack knowledge of more structured tools, such as the GNU Debugger (GDB). Here, they are introduced to tools they will need to use for debugging their own code in the elevator project.	[33]
6	Elevator project	This is the beginning of the course elevator project. The project culminates in a fully functional elevator control software suite, that is capable of handling arbitrary orders. In this first week, the students focus mainly on overall system design.	[4]
7	Elevator project	The students now have a decent understanding of the system requirements of the elevator controller. They have structured their ideas using UML diagrams to help communicate their design choices, and they are ready to start implementing.	[4]
8	Elevator project	The students begin coding their solutions. This is done in the C programming language, which is running on a computer connected to a model elevator with four floors, and buttons for “cab orders” as well as “external orders”.	[4]
9	Elevator project	The students freely use this time to code. Student assistants are present in the lab, to provide guidance to students who might need some input.	[4]
10	Elevator project	This is the last “dedicated week” of the elevator project. The students are free to continue work on their solution until the Factory Acceptance Test (FAT) in week 12, but this must be done outside of the normal lab hours.	[4]
11	micro:bit, build systems (make)	The students get acquainted with embedded systems by using the micro:bit platform. This week is dedicated to learning about General Purpose Input/Output (GPIO) in the form of buttons and LEDs. In addition to this, the students are introduced to automatic build systems, in this case GNU make, which is used further in the micro:bit labs.	[34]
12	micro:bit, Elevator FAT	This week is dedicated for full-duplex communication between embedded systems and host computers, by using UART (Universal Asynchronous Receiver Transmitter) peripherals. They also demonstrate their elevator implementation, which counts toward their final course score.	[35]
13	micro:bit	In this week, the students learn about low-power applications, by using the micro:bit nRF51822 “programmable peripheral interconnect” to directly couple buttons to tasks, such that the CPU does not have to be on.	[8]
14	micro:bit	Here, they learn about extending a one-chip system by using the I^2C (a.k.a. TWI) protocol to communicate with an accelerometer, and a magnetometer, present on the micro:bit platform.	[35]
15	micro:bit	This week, they use they accelerometer from last week to generate a pulse width modulated (PWM) signal, which drives a servomotor, based on what angle the students hold their micro:bit.	[35]

Fig. 5. Feedback collected from students regarding the helpfulness of conducting a code peer-review to evaluate code quality.

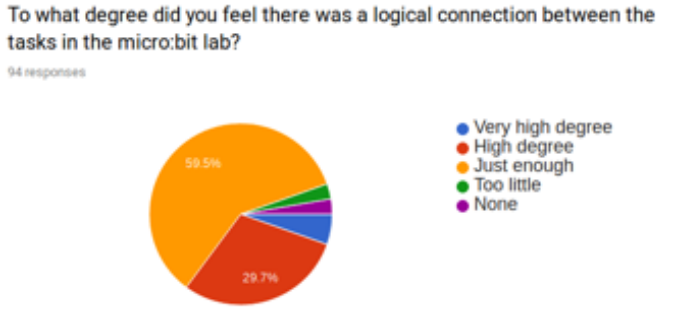

Fig. 6. Feedback from the students regarding the logical progression through the micro:bit exercises.

final grade the students achieve in the course. This way, the laboratory work feels meaningful, and the students have an extra incentive to fully absorb the concepts in the lab – rather than treating the lab work as something merely required to take the exam. To encourage a code quality standard [38], the students conducted a peer-review of other groups' code. The feedback collected suggests that this was one of the most beneficial aspects of the course, and it prompted increased awareness in code readability among the students, as shown in Figure 5.

Micro:bit C Programming. The micro:bit section of the laboratory is where the students get practical experience working with embedded devices – in this case the ARM Cortex®-M0 based nRF51822 system on chip (SoC) from Nordic Semiconductor [8]. This chip is embedded in the BBC micro:bit [2] platform. The micro:bit can be programmed in several ways. Out of the box it already supports an online JavaScript programming environment, as well as a version of microPython [2]. As this abstracts away a lot of the low-level details necessary for understanding the platform, the C programming language is selected instead for the proposed lab project. This approach allows the students to program the board nRF51822 SoC directly, while adding minimal overhead compared to the JavaScript and microPython approaches.

The labs occurring in the weeks 11-15 (see Table III) are first opened with an introduction to automatic build systems – in this case GNU Make [34] – which is used throughout the micro:bit labs. From there, the students advance through a

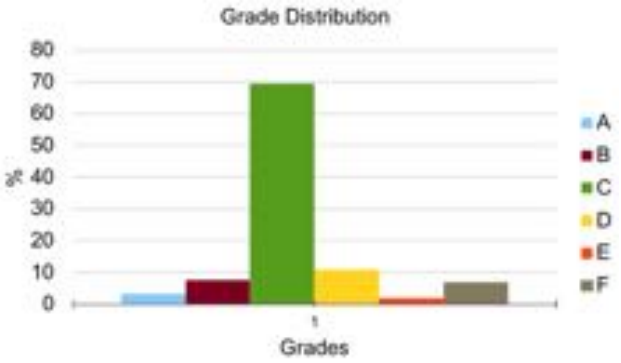

Fig. 7. The grade distribution for the students of the proposed embedded systems course.

number of common embedded systems applications, including General Purpose Input/Output (GPIO), Universal Asynchronous Transmitter-Receiver (UART), low power considerations with extended CPU sleep, the Inter-Integrated-Circuit (I2C) bus protocol, and Pulse Width Modulation (PWM) generation. All these tasks are supported by the content provided in parallel during th theoretical lectures of the course, as described in Sect. III. These labs are organized in such a way that the later labs build on the previous ones. For example, the PWM lab in week 15 (see Table III) uses the I2C-connected micro:bit accelerometer from week 14 (see Table III) to determine the desired pulse width. In this way, it is possible to avoid creating disjoint tasks that may feel meaningless on their own for the students. Based on the feedback from the students, this approach has been a good way to organize the lab content, as shown in Figure 6.

At the end of the course, the students are free to keep their micro:bit, and encouraged to experiment with them on their own. It remains to be seen how this will impact the retention of the course lab curriculum.

V. Course Lerning Outcomes

Portfolio evaluation is the basis for the final grade. Parts of the portfolio are final written exam (70%), exercises and laboratory work (30%). The result for each part is given in percentage units, while evaluation of the entire portfolio (the final grade) is given as a letter. The grades are A, B, C, D, and F, with A being the highest and F, short for failed, the lowest. The student grades distribution is shown in Figure 7. On a total number of 157 students, 3,2% achieved grade A, 7,6% concluded with grade B, 69,4% obtained grade C, 10,8% received grade D, 1,9% achieved grade E and 7% failed the exam.

VI. Conclusions and Future Work

In this work, a comprehensive syllabus of the embedded systems module for the engineering cybernetics education curriculum was presented. This module combines both a series of structured theoretical classes as well as practical and highly engaging laboratory assignments with group works. The students are involved with a highly-integrated organisation of the course, which includes system-oriented, hardware-oriented, software-oriented and application-oriented aspects of embedded systems. Theoretical notions are complemented with

a hands-on experience for implementing both industry standard embedded systems, such as Programmable Logic Controller (PLC) technology [1], as well as low-cost microcontrollers specifically designed for use in embedded systems education, such as the micro:bit microcontroller [2]. These choices contribute towards an application-ready, sustainable and effective educational experience. The analysis of results from student surveys indicates that the course organisation and topics are compelling and helpful.

In the future, the feedback received by the students can be consider improving their learning experience and the quality of the provided teaching offer. Further, this same educational approach can be applied to new modules for the engineering cybernetics education curriculum in combination with the use of open-source prototyping tools [39].

REFERENCES

[1] K. Collins, PLC programming for industrial automation. Exposure, 2007.

[2] G. Halfacree, "Getting started with the bbc micro: bit," The Official BBC micro: bitR User Guide, pp. 17–26.

[3] W. Rekdalsbakken and F. Sanfilippo, "Enhancing undergraduate research and learning methods on real-time processes by cooperating with maritime industries." in Proc. of the 28th European Conference on Modelling and Simulation (ECMS), Brescia, Italy, 2014, pp. 108– 114.

[4] Norwegian University of Science and Technology (NTNU). (2018, June) Course - Embedded Systems - TTK4235 - NTNU. [Online]. Available: https://www.ntnu.edu/studies/courses/TTK4235.

[5] M. Fowler, UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional, 2004.

[6] T. A. Henzinger and J. Sifakis, "The discipline of embedded systems design," Computer, vol. 40, no. 10, 2007.

[7] (2018, June) TTK4155 - Embedded and Industrial Computer Systems Design. [Online]. Available: https://www.ntnu.edu/studies/courses/TTK4155#tab=omEmnet

[8] Nordic Semiconductor. (2018, June) Nordic Semiconductor. [Online]. Available: www.nordicsemi.com/.

[9] L. Guo and R. Pecen, "Design projects in a programmable logic controller (plc) course in electrical engineering technology," in Proc. of the American Society for Engineering Education. Citeseer, 2008, pp. 1–10.

[10] F. Sanfilippo, O. L. Osen, and S. Alaliyat, "Recycling a discarded robotic arm for automation engineering education." in Proc. of the 28th European Conference on Modelling and Simulation (ECMS), Brescia, Italy, 2014, pp. 81–86.

[11] P. Marwedel, Embedded system design. Springer, 2006, vol. 1.

[12] C. Larman and V. R. Basili, "Iterative and incremental developments. a brief history," Computer, vol. 36, no. 6, pp. 47–56, 2003.

[13] B. Kernighan and D. M. Ritchie, The C programming language. Prentice hall, 2017.

[14] B. Tucker, "The flipped classroom," Education next, vol. 12, no. 1, pp. 82–83, 2012.

[15] P. Bradford, M. Porciello, N. Balkon, and D. Backus, "The blackboard learning system: The be all and end all in educational instruction?" Journal of Educational Technology Systems, vol. 35, no. 3, pp. 301–314, 2007.

[16] G. Bitar, "Programming of a FESTO production line," 2015, bachelor in Informatics and Automation (IA), Telemark University College, Norway.

[17] GNU Operating System. (2018, June) Making The Best Use of C. [Online]. Available: https://www.gnu.org/prep/standards/html_node/Writing-C.html.

[18] G. Martin, "Uml for embedded systems specification and design: motivation and overview," in Proc. of the Design, Automation and Test in Europe Conference and Exhibition. IEEE, 2002, pp. 773–775.

[19] L. Apvrille, P. de Saqui-Sannes, C. Lohr, P. Senac, and J.-P. Courtiat, ´ "A new uml profile for real-time system formal design and validation," in Proc. of the International Conference on the Unified Modeling Language. Springer, 2001, pp. 287–301.

[20] R. R. Yarlagadda, Analog and digital signals and systems. Springer, 2010, vol. 1.

[21] L. Frenzel, Principles of electronic communication systems. McGrawHill, Inc., 2007.

[22] SparkFun. (2018, June) Tutorials. [Online]. Available: https://learn. sparkfun.com/tutorials.

[23] M. Karlesky, G. Williams, W. Bereza, and M. Fletcher, "Mocking the embedded world: Test-driven development, continuous integration, and design patterns," in Proc. of the Embedded Systems Conference, CA, USA, 2007, pp. 1518–1532.

[24] S. Balaji and M. S. Murugaiyan, "Waterfall vs. v-model vs. agile: A comparative study on sdlc," International Journal of Information Technology and Business Management, vol. 2, no. 1, pp. 26–30, 2012.

[25] M. Siegel, "The sense-think-act paradigm revisited," in Proc. of the 1st International Workshop on Robotic Sensing (ROSE'03). IEEE, 2003, pp. 5–pp.

[26] T. Wescott, Applied control theory for embedded systems. Elsevier, 2011.

[27] I. Pan, S. Das, and A. Gupta, "Tuning of an optimal fuzzy pid controller with stochastic algorithms for networked control systems with random time delay," ISA transactions, vol. 50, no. 1, pp. 28–36, 2011.

[28] S. C. Bengea and R. A. DeCarlo, "Optimal control of switching systems," automatica, vol. 41, no. 1, pp. 11–27, 2005.

[29] G. K. McMillan, D. M. Considine et al., Process/industrial instruments and controls handbook. McGraw Hill, 1999, vol. 7.

[30] P. D. Pearson, M. L. Kamil, P. B. Mosenthal, R. Barr et al., Handbook of reading research. Routledge, 2016.

[31] H. Fairhead, "Micro: bit iot in c," 2016.

[32] git scm.com. (2018, June) Git version control system. [Online]. Available: https://git-scm.com/

[33] N. Matloff and P. J. Salzman, The Art of Debugging with GDB, DDD, and Eclipse. No Starch Press, 2008.

[34] F. S. Foundation, GNU Make Manual. Free Software Foundation, 2016.

[35] J. Catsoulis, Designing Embedded Hardware, 2nd ed. O'Reilly, 2005.

[36] H. Berger, Automating with STEP 7 in STL and SCL: SIMATIC S7-300/400 programmable controllers. John Wiley & Sons, 2014.

[37] D. T. Andrew Hunt, The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley Professional, 1999.

[38] S. McConnell, Code Complete, 2nd ed. Microsoft Press, 2004.

[39] F. Sanfilippo, H. Zhang, K. Y. Pettersen, G. Salvietti, and D. Prattichizzo, "ModGrasp: an open-source rapid-prototyping framework for designing low-cost sensorised modular hands," in Proc. of the 5th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob), Sao Paulo, Brazil ˜ , 2014, pp. 951–957.