Sustainable Approach to Teaching Embedded Systems with Hands-On Project-Based Visible Learning

Filippo Sanfilippo[1*†] **and Kolbjørn Austreng**[2]

[1]*Dept. of Engineering Sciences, University of Agder (UiA), Jon Lilletuns vei 9, Grimstad, 4879, Norway*

[2]*Dept. of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, 7491, Norway*

Abstract

Although purchasing state-of-the-art teaching equipment may be financially demanding, substantial efforts are being made at the Norwegian University of Science and Technology (NTNU) in Trondheim to provide students with an enhanced hands-on embedded system design experience in a sustainable manner. In particular, an approach that consists of adopting low-cost commercial off-the-shelf (COTS) components and tools for learning purposes is proposed in this work. This strategy effectively combines both industry standard highly-reliable automation controllers, such as *Programmable Logic Controller* (PLC) technology, as well as novel microcontrollers (i.e., the *micro:bit* microcontroller based on the *nRF51822* system-on-chip (SoC)) explicitly designed for use in embedded systems education. This contributes towards a hands-on sustainable learning experience based on the applicability of *Visible Learning* (VL). The objective of this paper is to propose a novel organisation of the embedded systems module for the engineering cybernetics education curriculum. The intended outcome is to promote a novel teaching approach. This is achieved by engaging students in both a series of organised theoretical lectures as well as practical and highly involving laboratory group projects. Surface learning sections and deep learning sections are thoroughly alternated to stimulate understanding, making relations, and extending the students' knowledge. The course organisation and main topics, as well as result analysis of student surveys are discussed. The survey results and feedback from the reference group indicate that the course organisation and topics are effective and helpful for the students.

Key words: embedded systems, education, programming, micro:bit

1 Introduction

Engineering cybernetics education is a multidisciplinary field of study that involves different types of knowledge and skills. This educational field is closely related to control theory and systems theory. This includes [1], among others: linear and nonlinear control

*Corresponding author.

†E-mail: filippo.sanfilippo@uia.no

theory, mathematical modelling, simulation, optimisation, embedded systems, real-time computer technology, and robotics.

In this paper, a novel organisation of the embedded systems module for the engineering cybernetics education curriculum is presented. The intended objective is to promote a novel way to teach the course. This work aims at answering the following research question: is it possible to stimulate understanding, trigger relations, and extend the students' knowledge by thoroughly alternating surface learning sections and deep learning sections? As shown in Figure 1, the fundamental 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) [2] and the advanced reduced instruction set computing (RISC) machine (ARM)-based embedded system *micro:bit* [3] jointly designed by the British Broadcasting Corporation (BBC), ARM and Nordic Semiconductor, 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 [4].

The presented course is *TTK4235 - Embedded Systems* [5]. 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 overall course organisation and main topics are presented in this paper. To show the effectiveness of the course from a pedagogical perspective, result analysis of pre-course and after-course surveys are outlined. The assessment of the survey results indicates that the module design and topics are engaging, effective and helpful for students.

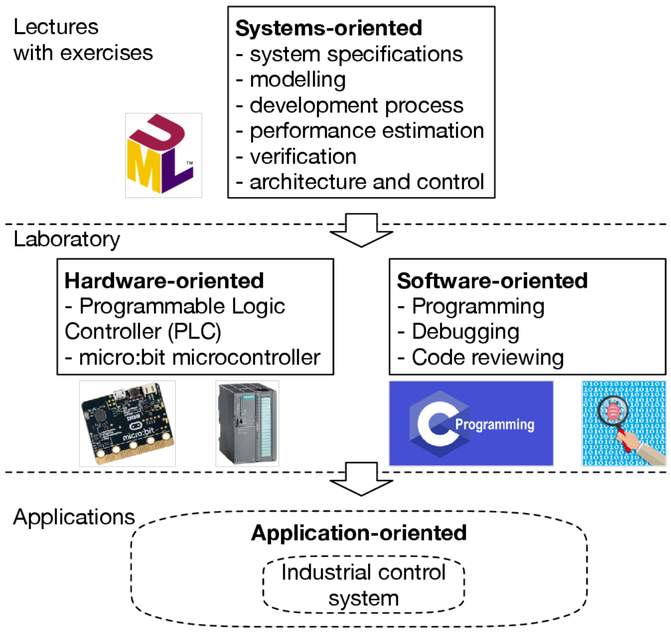

FIGURE 1: The proposed hands-on organisation of the embedded systems course.

The paper is organised as follows. In Section 2, an in-depth discussion on the selected pedagogical basis is presented. In Section 3, an overview of the chosen pedagogical tools is outlined. The course overview is discussed in Section 4. The laboratory content is depicted in Section 5. The course learning outcomes are delineated and analysed in Section 6. A discussion is presented in Section 7. Conclusions and future work are outlined in Section 8.

2 Pedagogical basis

In this section, an in-depth discussion of the selected pedagogical basis is presented. An embedded computing system uses microprocessors to implement portions of the functionality of non-general-purpose computers. Early microprocessor-based design courses, based on simple microprocessors, emphasised input and output (I/O) interfaces. Modern high-performance embedded processors are capable of a great deal of computation in addition to I/O tasks [6]. With the advent of novel microcontrollers, such as the *micro:bit* or the *Arduino* [7], embedded systems education requires updating the pedagogical bases and the corresponding content. In the recent literature, a review of embedded systems education in the *Arduino* age is presented in [8]. First, teaching challenges in embedded systems education are identified from the literature. Second, various *Arduino* teaching integration methodologies reported in the literature are surveyed and analysed. Third, the question whether *Arduino* successfully addresses embedded education challenges or not is discussed taking both surveyed findings and recent market trends into consideration. Regarding the *micro:bit*, successful education experiences are also reported in the literature. For instance, the impressions of students and teachers when they encountered the *micro:bit* in the classroom for the first time are reported in [9]. These previous works testify the validity of teaching with physical computing devices.

2.1 Challenges in meeting the needs of diverse learners

All educators embrace the commitment to help every student fulfil his or her full potential. This goal is even more challenging when considering technology instruction. Teachers frequently find students in a class showing a great deal of variety in needs and interests (i.e., enrolled in different study programmes but joining the same module). Students significantly vary in their motivation, prior knowledge and skills, learning styles, multiple intelligence, interests and backgrounds [10]. Because constructivist views of teaching and learning practices placing the learner's thinking or sense-making at the center of the teaching-learning process, teachers' attention to the incorporation of students' diversity of learning styles into teaching practice has become an increasingly relevant area for improvement [11]. Today's university students are diverse, not necessarily self-regulated, having varying skills in learning strategies and need to be deliberately taught [12].
There is an urgent need for a robust discipline about the scholarship of teaching and learning at the university level to best identify what works for the huge variety of studies on university learning. However, only few major syntheses exist [13].

2.2 John Hattie's *Visible Learning and Teaching*

By taking into consideration the need of diverse learners, John Hattie presented a unique and ground-breaking research on the applicability of *Visible Learning* (VL) to higher education in [12]. VL is a synthesis of more than 1200 meta-studies covering more than 80 million students. The aim of the synthesis is to place the various influences on student achievement along an underlying achievement continuum. According to John Hattie, VL is the result of 15 years of research. Based on John Hattie' experience, as nearly every intervention or method can show some evidence of success, we need to ask not "What works?" but "What works best" and seek comparisons between different ways of influencing student learning. VL is based on an enhanced role for teachers as they become evaluators of their own teaching. According to John Hattie VL occurs when teachers see learning through the eyes of students and help them become their own teachers [14]. To be successful, university teachers need to think of themselves as evaluators and ask about the merit, worth, and significance of the impact of their interventions – essentially, successful educators actively practice the *Scholarship of Teaching and Learning* (SoTL) [15].

2.3 Implications on surface and deep learning

The fundamental focus on impact requires questioning what "impact" means. To understand this implication, it is necessary to look at the effects on surface and deep learning [16]. According to literature, surface learning refers to students who complete the minimum tasks, memorise what is needed for an exam and nothing more. This is known as a surface approach [17] where students see learning tasks as enforced work. These students tend to be passive learners, working in isolation, and see learning as coping with tasks, so they can pass assessments. By contrast students who adopt a deep approach to learning will seek to understand meaning. They have an intrinsic interest and enjoyment in carrying out the learning tasks, and have a sincere curiosity in the subject, connections with other subjects and with building on their current learning. These students may enjoy social learning, including discussing different points of view. By considering these differences, a major finding from the VL research is that too often, surface learning is prioritised over deep learning. Several university teachers claim to their students that their course is about understanding, making relations and extending previous knowledge; but students see that the assessments in the course value knowing much and repeating back the major claims by the textbook or the lecturer [12]. Students are very strategic in identifying what teachers really value, as opposed to what they say they value. Based on these implications, the revision of the presented course (*TTK4235 - Embedded Systems* [5]) is focused on the analysis and review of the skills needed by the student to successfully answer the assignments and tests.

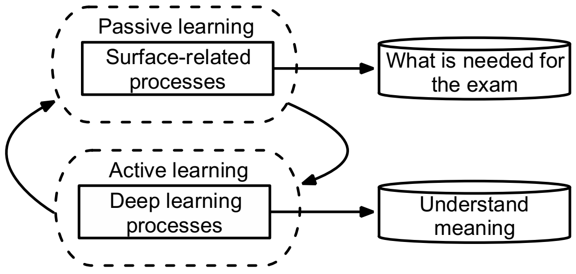

FIGURE 2: Surface-related versus deep learning processes.

Such a review was adopted as a powerful way to understand what impact looks like through the "eyes of students". Based on this impact, the presented course is organised by modulating and alternating surface-related teaching sections (i.e., exercises for the final exam) versus deep learning processes (i.e., laboratory tasks and assignments), as shown in Figure 2. Surface-related sections are adopted to comply with the module requirements in terms of formalities, grading and examination procedures. While, deep learning sections are emphasised and used to stimulate understanding, making relations, and extending the students' knowledge.

2.4 John Hattie's *Know Thy Impact*

The VL research identifies different major mind frames of successful teachers [12]. The most critical mind frame is "Know Thy Impact" – when an academic initiate a teaching session their crucial question needs to be "how will I know my impact today". This leads to the three sub-questions – "what do I mean by impact today and have I communicated this to my students, what is the magnitude of the impact I am seeking, and how many students can I teach such that they attain this magnitude on the impact I have clearly communicated". The other critical mind frames follow from this first "Know Thy Impact". The others include the following: "I am a change agent; I explicitly inform students what successful impact looks like from the outset; I see assessment as providing feedback about my impact; I work with other teachers to develop common conceptions of progress; I engage in dialogue not monologue; I strive for challenge and not 'doing your best'; I use the language of learning; and I see errors as opportunities for learning". Based on these major mind frames of successful teachers, the presented course (*TTK4235 - Embedded Systems* [5]) is based on a periodic feedback assessment of the impact from students through both short term as well as long term questionnaires and surveys (relevant results are presented in Section 5). Moreover, to ensure quality a reference group of students is established for the course. The reference group is made up of a subset of the course's students whose job is to constantly evaluate the course and write a report at the end [18]. The reference group should have an ongoing dialogue with other students throughout the course. The reference group may also choose to take other measures such as hold student meetings or conduct surveys before reference group meetings.

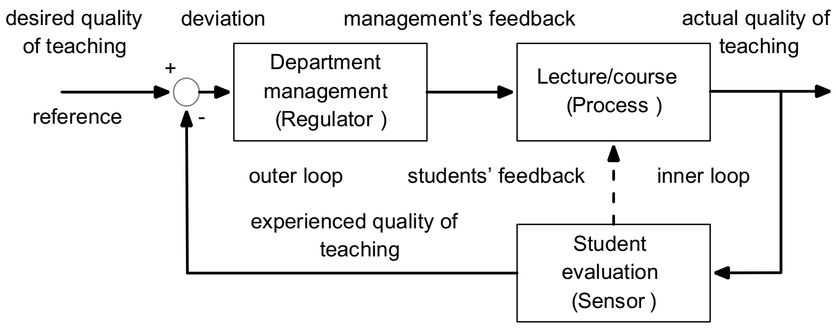
desired quality of teaching
reference
+
-
deviation
management's feedback
actual quality of teaching
Department management (Regulator)
Lecture/course (Process)
outer loop
students' feedback
inner loop
experienced quality of teaching
Student evaluation (Sensor)

FIGURE 3: To ensure quality a reference group of students is established for the course. A control feedback approach for teaching is adopted.

The reference group should represent all students enrolled in the course at the reference group meetings. As shown in Figure 3, a control feedback approach for teaching is adopted to ensure that the desired quality of teaching is achieved. The process to be controlled is the lecture/course. The reference group constantly provides constructive feedback based on the consistency between the learning outcomes, learning activities and the assessment. This feedback includes suggestions for changes that might help students achieve the learning outcomes for the course. This feedback is adopted for both an inner loop short-term iteration within the lecture/course, as well as for a long-term outer loop iteration with the department.

2.5 Achieving the highest possible impact through the *Visible Learning into Action*

The implementation model of VL is outlined more in detail into the *Visible Learning into Action* approach [19] by aiming at achieving the highest possible impact throughout the course. This approach is based on the idea of Clinton and Hattie [20] to explore much more in depth the notion of teachers as evaluators. This notion of educators as evaluators implies deliberate change, directing of learning, and visibly making a difference to the experiences and outcomes for the students (and for the teachers) – and the key mechanism for this activation is via a mind frame that embraces the role of evaluation. This mechanism is put into practice with a series of actions, such as factory acceptance tests (FAT) and peer reviewing processes, as discussed in Section 5.

2.6 Coordinated teaching

Coordinated teaching is the process of making connections between different topics of the same discipline or across separate courses. This is achieved by linking elements of the curriculum together across disciplines, but largely leaves teachers independent in their planning and instruction. A useful metaphor for the role of teachers, as highlighted in [21], is dancers at a sock hop. The overall theme of a curricular unit is like the music that keeps different dancers in rhythm. While they're moving to the same beat, teachers are on their own to determine the specific steps they'll perform. The music might also

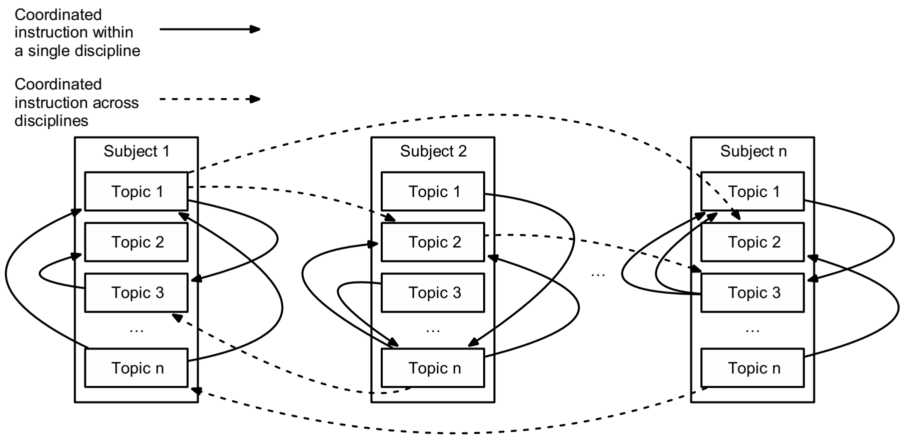

FIGURE 4: Coordinated instruction within a single discipline versus coordinated instruction across disciplines.

change regularly in an effort to please all the dancers; that is, teachers may take turns choosing the theme that best suits their subject goals. This idea is shown in Figure 4. The presented course (*TTK4235 - Embedded Systems* [5]), is developed by considering coordinated teaching across the entire engineering programme in cybernetics and robotics [1], as shown in Table 1.

2.7 Educating for innovation

On top of the necessary methodology for teaching efficiently and effectively, it is also necessary to contextualise and orient teaching activities from a socio-economic perspective. In the last several decades many of the world's most developed countries have shifted from an industrial economy to a knowledge economy, which is based on the creation of knowledge, information, and innovation.

Educational researchers have paid very little scholarly attention to this economic shift, although it has substantial implications [22]. In today's knowledge society, creativity always occurs in complex collaborative and organisational settings. For this reason, Sawyer [23] argues that education should be structured around disciplined improvisation, and he advocate the use of situated, collaborative knowledge-building activities. Sawyer also argues that creative collaboration in classrooms aligns with the social nature of innovation in today's economy. In line with this strategy, students of the course TTK4235 - Embedded Systems are divided into groups that stimulate their teamwork skills and critical thinking abilities.

3 Pedagogical tools

The unified modelling language (UML) [24] is adopted as a foundation for the proposed module. This choice is motivated by the fact that UML can potentially be utilised to

TABLE 1
Engineering programme in cybernetics and robotics [1].

Semester	Modules			
10th semester (spring)	Thesis			
9th semester (fall)	Complementary topic	Specialisation course	Specialisation project	
8th semester (spring)	Experts in Teamwork (EiT)	Selective topic	Selective topic	Selective topic
7th semester (fall)	Complementary topic	Selective topic	Selective topic	Selective topic
6th semester (spring)	Selective topic	Real-time Programming	Modelling and Simulation	Optimisation and Control
5th semester (fall)	Selective topic	Technology Management	Algorithms and Data Structures	Linear System Theory
4th semester (spring)	Statistics	Exam for Science and Technology	Control Systems	Embedded Systems
3rd semester (fall)	Mathematics 4	Physics	Computers and Digital Design	Industrial Electrotechnics
2nd semester (spring)	Mathematics 2	Mathematics 3	Procedural and Object-Oriented Programming	Instrumentation and Measurements
1st semester (fall)	Mathematics 1	Information Technology, Introduction	Electrical Circuits and Digital Design	Cybernetics Introduction

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 [25].

The choice of the BBC *micro:bit* platform [3] 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 [26]. This requirement is met by the Nordic Semiconductor nRF51822 System on Chip [27], which is the main component of the BBC *micro:bit* platform. Furthermore, for practical and logistical reasons, it is desirable to use a more or less “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

TABLE 2

The organization of the course content.

Lectures	Laboratory	Projects
8 lectures	15 laboratories	elevator project (PLC) elevator project (*C*-programming) *micro:bit* (*C*-programming)

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 BBC *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*++ [3]. 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 [28], a cost-effective approach for the design of educational projects in a PLC course for electrical engineering education is presented. In [29], 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.

4 Course overview

In this section, the proposed course organisation and main topics are presented [30]. As shown in Table 2, the course content includes 8 theoretical lectures, laboratory classes and 3 course projects. The content of the theoretical lectures is depicted in Table 3. 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 [5]. Successively, an introduction of embedded system with the most relevant descriptions, definitions and vocabulary is considered [31]. 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 a set of jumper wires. 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 [32]. Finally, an introduction of UML class diagrams is given [24].

Lecture 2: flipped class on *C*-programming, PLC introduction

Lecture 2 consists of a review of the main topics of the *C* programming language [33]. 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 [34] 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 [35], 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) [2] 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 [36].

Lecture 3: UML concepts and class exercise

Lecture 3 introduces the fundamental concepts of UML sequence diagrams [24, 37, 38] 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, for illustrating the focus of control. The notion of message (or stimulus) and of lifeline is 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, 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 [24]. 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 superstates. 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. Further, UML activity diagrams are introduced as a technique to describe procedural logic, business process, and workflow. These diagrams present several similarities to flowcharts, but the principal difference between them and the flowchart notation is that they support parallel behaviour [24]. Additionally, UML communication diagrams are presented as a tool to identify interactions between objects and/or components (represented as lifelines) of the system to be modelled using sequenced messages in a free-form arrangement. 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 [39-41]. 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 in this lecture 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 [41]. 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) [40, 41]. To prepare the students to the forthcoming laboratory work, a review of code verification techniques for C-programming is finally presented [42].

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 [40]. 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 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 [40]. 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 [40]. 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* [43] 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 [44]. This requires 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 [45]. 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 [46] and a bang–bang controller (2 step or on–off controller) [47], 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 [48] 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 [49]. 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.

5 Laboratories overview

In this section, the proposed laboratory organisation and main topics are presented, as shown in Table 4. The laboratory content is run in parallel with the theoretical lectures presented in Section 4. In the following of this section, the main laboratory topics are presented referring to Table 4.

PLC programming

The PLC laboratory 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 seen in Figure 5. The PLC is then programmed using the SIMATIC *STEP-7* software [55]. As there are good manuals available for this [55], 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, both regarding understanding of the topic, as shown in Figure 6, as well as in terms of further engagement, as shown in Figure 7. 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 programming the elevator setup in the C programming language (see Section 5). 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 [51]. 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 [56].

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 [56]. To this end, they are introduced to the GNU Debugger (GDB) [52]. Students are also introduced to Valgrind [57] for memory-related issues such as segmentation faults and memory leaks, which GDB is not as well suited for. Valgrind is an instrumentation framework for building dynamic analysis tools. There are Valgrind tools that can automatically detect many memory management and threading bugs, and profile programs in detail.

TABLE 3

The organisation of the course theoretical lectures.

Week	Lectures	Description	References	Time
1	1	- Course overview, expectations, logistics, processes, syllabus, FAQ, and prerequisite material	[5]	2 hours
		- Embedded systems descriptions, definitions and vocabulary - Microprocessor/microcontroller architectures - *micro:bit* introduction: Nordic Semiconductor Seminar	[3, 31, 50]	2 hours
		- UML introduction; paradigms and models (development process): prototyping, incremental, waterfall - UML: Class Diagrams	[24, 32]	2 hours
3	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	[33]	5 hours
		- PLC introduction: "Programming of a FESTO production line"	[2]	0.5 hours
		- *C*-programming: coding style, making the best use of *C*	[36]	0.5 hour
5	3	- UML Sequence Diagrams, Use cases, Class Diagrams, advanced concepts	[24, 37, 38]	4 hours
		- Class exercise: Use cases, Class Diagram and Sequence Diagrams for the Elevator project		2 hours
7	4	- UML: State Machines, activity, communication and Timing Diagrams	[24]	4 hours
		- Class exercises: State Machines and Timing Diagrams for the Elevator project		2 hours
9	5	- Analog and digital signals	[39, 40]	1 hour
		- Communication protocols: serial, UART, SPI, I2C	[41, 50]	4 hours
		- C-programming: code verification	[42]	1 hour
11	6	- ADC and DAC, aliasing	[40]	1 hour
		- Modulation (AM, FM, PWM)	[40]	2 hours
		- Software Development Cycle: non-functional Vs functional requirements, Waterfall Vs Iterative V Model	[24]	1 hours
		- Class exercises: UML modelling of an automated teller machine (ATM)		2 hours
14	7	- Industrial instrumentation and control		2 hours
		- Temperature control	[48]	1 hour
		- Error in measurements		1 hour
		- Class exercises: UML modelling of a Vending Machine		2 hours
16	8	- Reading research papers	[49]	2 hours
		- Exam simulation: requirements analysis, development process selection, system design, sensors		4 hours

TABLE 4

The organization 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.	[51]
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.	[52]
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.	[5]
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.	[5]
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".	[5]
9	Elevator project	The students freely use this time to code. Student assistants are present on the lab, to provide guidance to students who might need some input.	[5]
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.	[5]
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.	[53]
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.	[54]

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.	[27]
14	*micro:bit*	Here, they learn about extending a one-chip system by using the I2C (a.k.a. TWI) protocol to communicate with an accelerometer, and a magnetometer, present on the micro:bit platform.	[54]
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*.	[54]

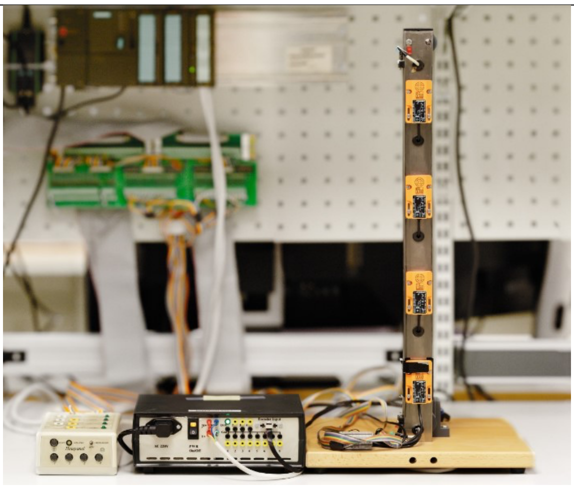

FIGURE 5: The elevator model setup used in the course project. Courtesy of the Dept. of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

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 5. 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 headfirst 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 4. 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 firsthand in a laboratory project where they are needed, it is very beneficial for the retention of the material - as the class feedback has shown.

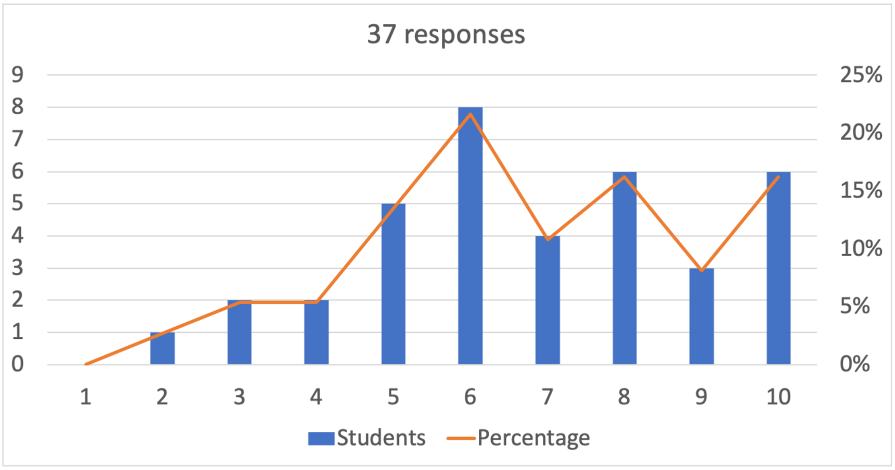

FIGURE 6: Feedback collected from students regarding the helpfulness of conducting the PLC programming lab concerning understanding of the topic (on a scale from 1 to 10).

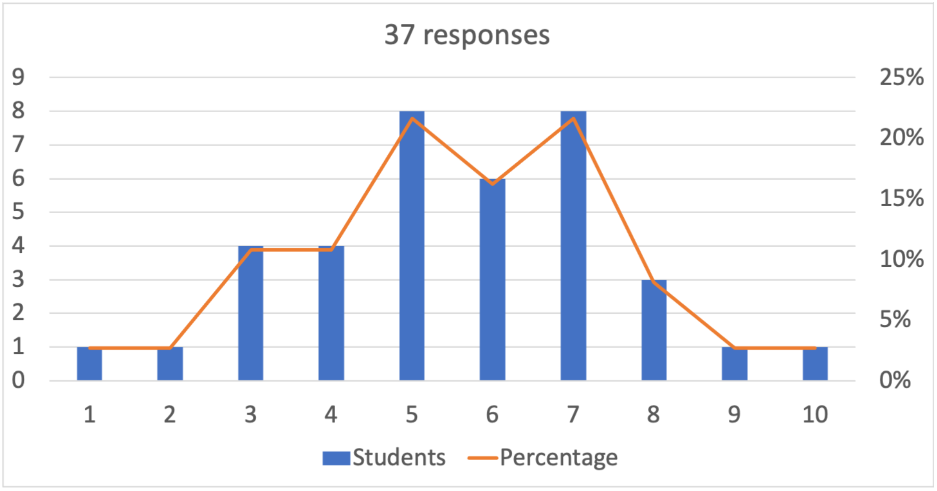

FIGURE 7: Feedback collected from students regarding the helpfulness of conducting the PLC programming lab regarding further engagement (on a scale from 1 to 10).

Elevator factory acceptance test (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 4. This acceptance test is graded and counts toward the 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 [58], 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 8.

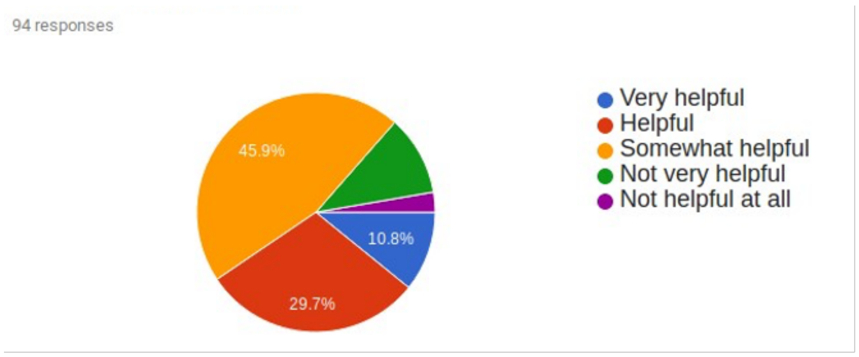

FIGURE 8: Feedback collected from students regarding the helpfulness of conducting a code peer-review to evaluate code quality.

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 [27]. This chip is embedded in the BBC *micro:bit* [3] 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* [3]. 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 Table4) are first opened with an introduction to automatic build systems - in this case GNU Make [53] - which is used throughout the *micro:bit* labs. From there, the students' progress through a 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 (I^2C) bus protocol, and Pulse Width Modulation (PWM) generation. All these tasks are supported by the content provided in parallel during the theoretical lectures of the course, as described in Section 4.

These labs are organised in such a way that the later labs build on the previous ones. For example, the PWM lab in week 15 (see Table 4) uses the I^2C-connected *micro:bit* accelerometer from week 14 (see Table 4) 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 organise the lab content, as shown in Figure 9.

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.

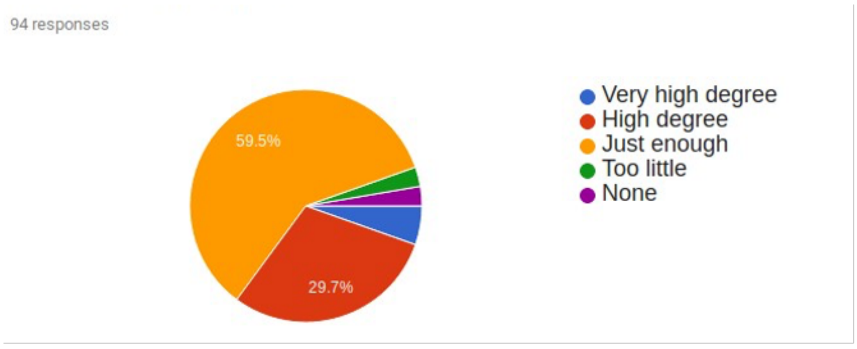

FIGURE 9: Feedback from the students regarding the logical progression through the *micro:bit* exercises.

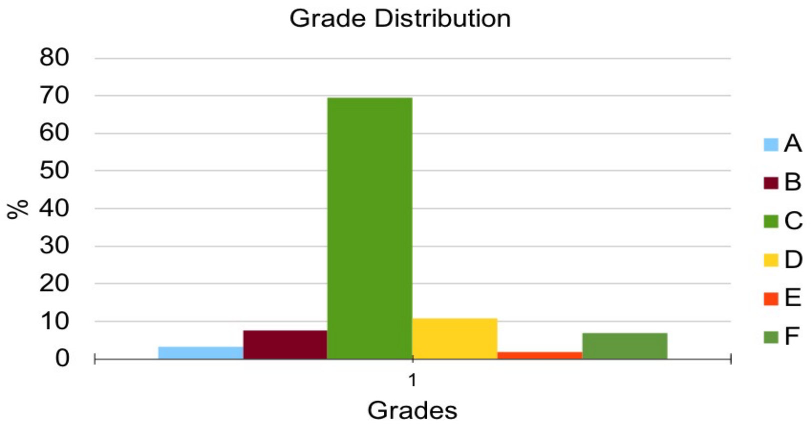

FIGURE 10: The grade distribution for the students of the embedded systems course.

6 Course learning outcomes and feedback from the reference group

Portfolio evaluation is the basis for the final grade. Parts of the portfolio are final written exam (70%) and exercise 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 10. 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.

The reference group provided a detailed report regarding the course outcomes. The following aspects were described:

- during the course, logical connections for coordinated instruction across disciplines were provided to the students, i.e., modelling, programming, design, physics, research methods and mathematics. In the future, more effort shall be put on acquiring a more accurate knowledge of the entire study programme so that instruction across disciplines can be tailored even more effectively through more accurate coordinated teaching actions;

- during the course, the students really appreciated the laboratory work as being educational, challenging and fun. In the future, the lecture time dedicated to deep learning processes (i.e., laboratory work and assignments) shall be increased even more;
- the assignments were setup to be done during lab hours and they felt connected to the labs, according to the students' feedback;
- to improve the students' involvement, the interaction between the teacher and the students shall be maximised in the future. This interaction has an important role in the teaching and learning process and, therefore, it is vital for the classroom activities;
- the dialogue with industrial partners should also be maximised to promote a lifelong learning curricula based on university-industry synergic approach [59].

7 Discussion

Based on the presented results, this work contributes towards filling up the major gap in understanding the benefits of applying a hands-on sustainable learning experience based on the principle of Visible Learning (VL) for embedded systems education. The proposed novel organisation of the embedded systems module considers a balanced combination of theoretical lectures, practical laboratories and applications. From a pedagogical perspective, the presented approach takes into account the need of diverse learners by engaging students in highly involving group projects. The overall methodology outlined in this work may also be applicable to other field of studies. The adopted methodology to studying the gap is based on surveys and feedback received from students. The results presented are in line with similar studies in the field. Potential limitations in the design of the presented study may concerns the sample size as well as the distribution and diversity of students.

8 Conclusions and Future Work

A comprehensive syllabus of the embedded systems module for the curriculum of engineering cybernetics education was introduced in this work. The following research question was considered: is it possible to stimulate understanding, trigger relations, and extend the students' knowledge by thoroughly alternating surface learning sections and deep learning sections? The presented module combines both a series of structured theoretical classes as well as practical and highly engaging laboratory assignments with group works. The students are engaged in a highly interconnected organisation of the course, which incorporates system-oriented, hardware-oriented, software oriented and application-oriented aspects of embedded systems. Surface learning sections and deep learning sections are systematically alternated to promote comprehension, interaction and broadening of the students' knowledge. To achieve this, theoretical principles are accompanied with hands-on experience for implementing both industry standard embedded systems, such as *Programmable Logic Controller* (PLC) technology, as well as low-cost microcontrollers purposely designed for use in embedded systems education, such as the *micro:bit* microcontroller. Throughout the course, logical connections for coordinated instruction across disciplines are provided to the students, i.e., modelling, programming, design, physics, research methods and mathematics. These choices contribute towards a hands-on sustainable learning experience based on the applicability of *Visible Learning* (VL). The analysis of results from student surveys and feedback from the reference group indicates that the course organisation and topics are compelling and helpful.

In the future, the amount of PLC coverage might be reduced, as they are reaching end-of-life, and students have no difficulties grasping the concepts either way. Field programmable gate arrays (FPGAs) might be introduced instead, as the industry is becoming "more embedded". Hence, the course should reflect this market trend. From a pedagogical perspective, the feedback received by the students can be considered to improve 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.

References

1. Norwegian University of Science and Technology (NTNU), "Master engineering programme in cybernetics and robotics," February 2019. [Online]. Available: https://www.ntnu.no/studier/mttk/oppbygning
2. K. Collins, *PLC programming for industrial automation*. Exposure, 2007.
3. G. Halfacree, "Getting started with the bbc micro: bit," *The Official BBC micro: bit® User Guide*, pp. 17–26, 2019.
4. 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.
5. Norwegian University of Science and Technology (NTNU), "Course Embedded Systems - TTK4235 - NTNU," June 2018. [Online]. Available: https://www.ntnu.edu/studies/courses/TTK4235.
6. W. Wolf and J. Madsen, "Embedded systems education for the future," *Proceedings of the IEEE*, vol. 88, no. 1, pp. 23–30, 2000.
7. Arduino, "Arduino, an open-source electronics prototyping platform," December 2020. [Online]. Available: http://arduino.cc/.
8. M. El-Abd, "A review of embedded systems education in the Arduino age: lessons learned and future directions," 2017.
9. M. Videnovik, E. Zdravevski, P. Lameski, and V. Trajkovik, "The BBC micro: bit in the international classroom: learning experiences and first impressions," in *Proc. of the IEEE 17th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 2018, pp. 1–5.
10. O. N. Ukpokodu, "Meeting the needs of diverse learners," *Social Studies and the Young Learner*, vol. 16, no. 1, pp. 31–32, 2003.
11. J. Biggs, "Constructive alignment in university teaching," *HERDSA Review of higher education*, vol. 1, no. 5, pp. 5–22, 2014.
12. J. Hattie, "The applicability of visible learning to higher education." *Scholarship of Teaching and Learning in Psychology*, vol. 1, no. 1, p. 79, 2015.

13. E. T. Pascarella and P. T. Terenzini, “How college affects students: A third decade of research (vol. 2),” 2005.
14. Sebastian Waack, “Visible Learning,” December 2020. [Online]. Available: https://visible-learning.org/
15. M. T. Huber, *The advancement of learning: Building the teaching commons*. Wiley, 2005.
16. T. W. Smith and S. A. Colby, “Teaching for deep learning,” *The Clearing House: A Journal of Educational Strategies, Issues and Ideas*, vol. 80, no. 5, pp. 205–210, 2007.
17. D. H. Dolmans, S. M. Loyens, H. Marcq, and D. Gijbels, “Deep and surface learning in problem-based learning: a review of the literature,” *Advances in health sciences education*, vol. 21, no. 5, pp. 1087–1112, 2016.
18. Norwegian University of Science and Technology (NTNU), “Reference groups - quality assurance of education,” February 2019. [Online]. Available: https://innsida.ntnu.no/wiki/-/wiki/English/Reference+groups+-+ quality+assurance+of+education
19. D. Masters, K. Birch, and J. Hattie, *Visible learning into action: International case studies of impact*. Routledge, 2015.
20. J. Hattie and J. Clinton, “School leaders as evaluators,” *Activate: A leader’s guide to people, practices and processes*, pp. 93–118, 2011.
21. Discovery Education, “Know Your Education: Conceptual Differences Between Integrated and Coordinated Instruction,” February 2019. [Online]. Available: http://frontandcentral.com/teaching-and-learning/know-educonceptual- differences-integrated-coordinated-instruction/
22. R. K. Sawyer, “Educating for innovation,” *Thinking skills and creativity*, vol. 1, no. 1, pp. 41–48, 2006.
23. R. K. Sawyer, *Explaining creativity: The science of human innovation*. Oxford university press, 2011.
24. M. Fowler, *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004.
25. T. A. Henzinger and J. Sifakis, “The discipline of embedded systems design,” *Computer*, vol. 40, no. 10, 2007.
26. “TTK4155 - Embedded and Industrial Computer Systems Design,” June 2018. [Online]. Available:
    https://www.ntnu.edu/studies/courses/TTK4155tab=omEmnet
27. Nordic Semiconductor, “Nordic Semiconductor,” December 2020. [Online]. Available: www.nordicsemi.com/.
28. 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.
29. 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.
30. F. Sanfilippo and K. Austreng, “Enhancing teaching methods on embedded systems with project-based learning,” in *Proc. of the IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 2018, pp. 169– 176.
31. P. Marwedel, *Embedded system design*. Springer, 2006, vol. 1.

32. C. Larman and V. R. Basili, “Iterative and incremental developments. a brief history,” *Computer*, vol. 36, no. 6, pp. 47–56, 2003.
33. B. Kernighan and D. M. Ritchie, *The C programming language*. Prentice hall, 2017.
34. B. Tucker, “The flipped classroom,” *Education next*, vol. 12, no. 1, pp. 82–83, 2012.
35. 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.
36. GNU Operating System, “Making the Best Use of C,” April 2020. [Online]. Available: https://www.gnu.org/prep/standards/html node/Writing-C.html.
37. 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.
38. R. Ahmadi, E. Posse, and J. Dingel, “Slicing uml-based models of real-time embedded systems,” in *Proc. of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 2018, pp. 346–356.
39. R. R. Yarlagadda, *Analog and digital signals and systems*. Springer, 2010, vol. 1.
40. L. Frenzel, *Principles of electronic communication systems*. McGraw-Hill, Inc., 2007.
41. SparkFun, “Tutorials,” December 2020. [Online]. Available: https://learn.sparkfun.com/tutorials.
42. 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.
43. 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.
44. 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.
45. T. Wescott, *Applied control theory for embedded systems*. Elsevier, 2011.
46. 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.
47. Y. Wang, B. Xiao, L. Liu, and Q. Yuan, “Bangbang controller design and implementation for east vertical instability control,” *Fusion Engineering and Design*, vol. 112, pp. 692–698, 2016.
48. G. K. McMillan, D. M. Considine *et al.*, *Process/industrial instruments and controls handbook*. McGraw Hill, 1999, vol. 7.
49. P. D. Pearson, M. L. Kamil, P. B. Mosenthal, R. Barr *et al.*, *Handbook of reading research*. Routledge, 2016.
50. H. Fairhead, *Micro: bit IoT In C*. I/O Press, 2016.
51. git scm.com, “Git version control system,” December 2020. [Online]. Available: https://git-scm.com/
52. N. Matloff and P. J. Salzman, *The Art of Debugging with GDB, DDD, and Eclipse*. No Starch Press, 2008.
53. F. S. Foundation, *GNU Make Manual*. Free Software Foundation, 2016.
54. J. Catsoulis, *Designing Embedded Hardware*, 2nd ed. O’Reilly, 2005.
55. H. Berger, *Automating with STEP 7 in STL and SCL: SIMATIC S7-300/400 programmable controllers*. John Wiley & Sons, 2014.

56. D. Nadeau, N. Ezzati-Jivan, and M. R. Dagenais, “Efficient large-scale heterogeneous debugging using dynamic tracing,” *Journal of Systems Architecture*, vol. 98, pp. 346–360, 2019.
57. Valgrind Developers, “Valgrind, a suite of tools for debugging and profiling,” December 2020. [Online]. Available: https://valgrind.org/

58. S. McConnell, *Code Complete*, 2nd ed. Microsoft Press, 2004.

59. F. Falcone, A. V. Alejos, J. G. Cenoz, and A. L. Mart´ın, “Implementation of higher education and life long learning curricula based on university-industry synergic approach,” *The International journal of engineering education*, vol. 35, no. 6, pp. 1568–1583, 2019.

Author biography

Filippo Sanfilippo an Associate Professor at the Dept. of Engineering Sciences, Faculty of Engineering and Science, University of Agder (UiA), Grimstad, Norway. He is also appointed as a Professor II at the Dept. of Mechanical, Electronic and Chemical Engineering, Faculty of Technology, Art and Design, Oslo Metropolitan University (OsloMet), Oslo, Norway. He is responsible for the research theme on Collaborative Robots (CoBots) at the Priority Research Centre Mechatronics, UiA. He is also a member of the Centre for Integrated Emergency Mangement (CIEM), UiA. He holds a PhD degree in Engineering Cybernetics. His research focuses on robotics, wearable haptics and safe human-robot interaction. He is currently supervising four PhDs and co-supervising one PostDoc. He carries experience in participating to European research programs and various national projects from the Research Council of Norway (RCN). He is an IEEE Senior Member. He is currently the Chair of the IEEE Norway Section. He is also the Chair of the IEEE Robotics and Automation, Control Systems and Intelligent Transportation Systems Joint Chapter. He is also the treasurer of the Norsk Forening for Kunstig Intelligens (NAIS), the Norwegian Association for Artificial Intelligence. He has authored and co-authored several technical papers in various journals and conferences. He is a reviewer for several international conferences and journals.

Kolbjørn Austreng holds a master’s degree in cybernetics and robotics from the Norwegian University of Science and Technology (NTNU), Norway. He actively contributes to the course TTK4235 - Embedded Systems as a Teaching Assistant.
He has also been an active member of Revolve NTNU, where he was involved in the development of embedded sensor-and telemetry circuits, with a cross disciplinary focus on both hardware and software. He is currently an active alumnus of Revolve NTNU. He has also been involved in the development of internal tools at Nordic Semiconductor, with a focus on performance embedded systems testing.