An Open Framework for teaching Motion Control for Mechatronics Education

Filippo Sanfilippo*, Martin Økter, Tine Eie and Morten Ottestad
Dept. of Engineering Sciences, University of Agder (UiA), Jon Lilletuns vei 9, 4879, Grimstad, Norway
*Corresponding author: filippo.sanfilippo@uia.no

***Abstract*—This paper proposes the introduction of a novel open prototyping framework that involves using low-cost commercial off-the-shelf (COTS) components and tools for the module of motion control, within mechatronics education. The goal of this study is to propose a novel structure for the motion control module in the engineering mechatronics curriculum. This is accomplished by integrating students in a series of well-organised theoretical lectures as well as hands-on, highly engaging laboratory group projects. Surface learning parts and deep learning sections are combined frequently to encourage learners to grasp, make connections, and expand their knowledge. The structure of the course as well as the key topics are discussed. The proposed open framework, which consist of an elevator model, is presented in details.**

***Index Terms*—education, mechatronics, hands-on learning, framework.**

I. Introduction

Motion control is a branch of automation that encompasses the systems and subsystems involved in the controlled movement of machine parts. Precision engineering, micromanufacturing, biotechnology, and nanotechnology are just a few of the fields where motion control systems are widely used for automation [1]. The key components involved with motion control usually include an energy amplifier, and one or more prime movers or actuators. There are two types of motion control: open loop and closed loop. The main focus of motion control is the specific automation control of motion systems with electric actuators such as DC/AC servo motors. Control of robotic manipulators is also considered as part of the field of motion control because the majority of robotic manipulators are driven by electrical servo motors and the primary goal is the control of motion.

In this work, a novel organisation of the *motion control* module for the engineering mechatronics education curriculum is presented. The aim is to inspire a new approach to teaching the course. Inspired by our previous work [2], [3], [4], this study intends to address the following research question: can thoroughly alternating surface and deep learning sessions increase understanding, activate relationships, and enhance students' knowledge? The basic idea is to organise the course into three parallel levels: (i) lectures that combine theory and exercises; (ii) laboratory; (iii) applications, in particular, an elevator model is prototyped.

The presented course is *MAS246-G* [5]. This course is a 5th semester module of the three years bachelor's degree programme in Mechatronics [6] given at the Department of Engineering Sciences, Faculty of Engineering and Science, University of Agder (UiA), Grimstad, Norway. Recommended previous knowledge for the course includes *MAS239* Feedback Control Systems 1, *MAS134* Electrical circuits and digital engineering, *MA178* Mathematics 1, *MA-179* Mathematics 2, or equivalent. This article outlines the overall structure of the course as well as the primary themes.

The paper is organised as follows. Section II depicts the course overview, while the laboratories are presented in Section III. Section IV describes the selected elevator model. The proposed architecture for the elevator model is presented in Section V. Finally, Section VI contains the conclusions and recommendations for further studies.

II. Course overview

The course material includes 12 theoretical lectures, 12 laboratory classes, and one course project. Each 6-hour theoretical lecture is held once a week and followed by a weekly 6-hour laboratory session. The following is a list of the topics covered in each lecture:

- Lecture 1: introduction on direct current (DC) machines;
- Lecture 2: DC machine's various drive circuits and operations;
- Lecture 3: modelling of a DC motor as thermal system;
- Lecture 4: stepper motor types and working principles;
- Lecture 5: control methods for stepper motors;
- Lecture 6: brushless DC electric motors;
- Lecture 7: permanent-magnet synchronous motors (PMSM);
- Lecture 8: rotary to rotary motion transmissions;
- Lecture 9: rotary to translational motion transmissions;
- Lecture 10: shaft selection and sizing;
- Lecture 11: lead-lag compensators;
- Lecture 12: modern motion control architecture.

III. Laboratories overview

The laboratory sessions run in parallel with the theoretical lectures introduced in Sect. II.

Programming in practice: Arduino

The goal of this lab is to introduce the use of Arduino [7] as a developing platform for motion control. Motion control necessitates a number of skills and abilities that can be easily developed using tools such as Arduino.

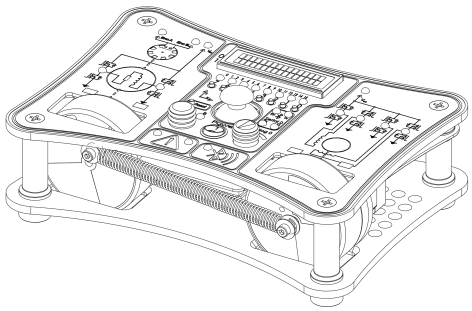

Fig. 1: The all in one servo lab (AIOSL).

Unified modelling language (UML) and class exercise

An introduction of the Unified modelling language (UML) is given with particular emphasis on embedded systems. Different diagrams are introduced including use case diagrams, class diagrams, sequence diagrams, and state machine diagrams [8]. These diagrams are meant to support the design and development of the course project.

All in one servo lab (AIOSL)

To facilitate the learning process for students, we have designed a novel platform for doing practical laboratory assignments, simulations and demonstrations. The development of this platform is based on the following key requirements: i) ease of use, plug and play (PnP), and ii) students engagement, so that working with the platform should be fun. To meet these requirements, we have developed a single unit platform with servo and stepper motor, various inputs and feedback options. This platform is named "all in one servo lab" (AIOSL) [9] and it is shown in Figure 1. Among other components, the AIOSL embeds two motors: a brushed DC motor and a two-phase stepper motor, respectively. This allows students to develop multi-function applications. The DC motor gives students the possibility to design software for motor control and to utilise lead-lag regulation. For closed loop control, feedback from the embedded encoder is necessary. This application challenges the students' ability to regulate micro-stepping. The components of the AIOSL are integrated in the chassis, while also providing a functional design. In fact, the motors are exposed through the top panel, giving a more visual perception of their status and allowing the students to affect their flywheels.

From a hardware perspective, an Arduino Mega [7] is integrated into the AIOSL.

IV. Elevator model

The course project aims at designing a complete elevator system to be implemented based on the AIOSL.

A. System requirements

As shown in Figure 2, the system controls a single-cabin elevator that travels up and down in a building with a set number of floors. The system requirements are summarised in Table I [10]. Based on these system requirements, the system features are mapped to the hardware and software components.

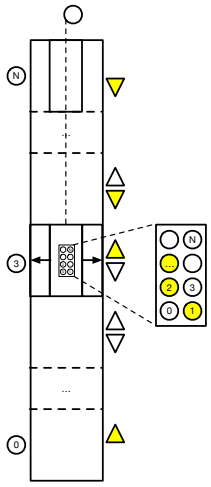

Fig. 2: An elevator system model with 0 to N floors, including a car, cables, an elevator machine, controls drive, cabin buttons and floor buttons.

TABLE I: System requirements.

REQ 1	The system controls the movement of an elevator.
REQ 2	The number of floors is set.
REQ 3	The elevator is either going up or going down.
REQ 4	The elevator is driven by a motor which can be either WINDING, UNWINDING, or STOPPED.
REQ 5	If not at the top floor, the cabin moves up one floor if the motor is WINDING
REQ 6	If not at the bottom floor, the cabin moves down one floor if the motor is UNWINDING
REQ 7	The cabin has a door which can be OPEN, HALF, or CLOSED.
REQ 8	While the cabin is moving, its door must be closed.
REQ 9	On each floor except the top one, there is an "up" button.
REQ 10	On each floor except the bottom one, there is a "down" button.
REQ 11	Inside the cabin, there are floor buttons, one for each floor.
REQ 12	The cabin stops at a particular floor and opens the door if there is a request to serve at that particular floor.
REQ 13	The requests at one floor are cleared once the door is fully open.
REQ 14	The elevator should not move to leave a floor if there are requests to serve at that floor.
REQ 15	The elevator should stay stationary at a floor when there are no requests.
REQ 16	The elevator can only change direction if it has no requests in the same direction, but has some requests in the opposite direction.

Elevation maps the requirements of WINDING, UNWINDING. Physical doors are implemented to more closely simulate a real elevator. There is an external cabin caller, and floor buttons inside the cabin. The model is also equipped with a floor indicator to visualise the current position. Although not strictly required, the elevator also includes a path optimiser for an efficient management of the elevator queue.

V. Proposed Architecture

A. Hardware design

For the elevator to take use of its functions, students are engaged in mapping these features into hardware components.

TABLE II: Mapping table for elevator model.

Feature	Hardware		Software
	Sensors	*Actuators*	
Elevation	Ground floor switch	Stepper with threaded rod	
Doors		Servo	
Floor buttons inside cabin	Button (7-2)		
Floor indicator		LCD-Screen	
Cabin-caller	Potentiometer for selection, buttons (1-0) for up and down	LCD-Screen	
Path-optimizer			Queueing system

For elevation (WINDING, UNWINDING), a stepper motor with a threaded rod is utilised. A mechanical switch indicates when the elevator hits the ground floor. For the doors, a servo motor is adopted. The floor buttons have sensors corresponding to each floor. The cabin caller is equipped with a potentiometer to externally select the floor, and with an LCD screen, also for floor indication. Even though a mapping to hardware components is prioritised, some features may alternatively be implemented in software. For example, the path optimiser is ruled by a queuing system implemented via software. The overall mapping of the elevator features is shown in Table II.

Referring to Figure 3, wires are connected in the stand to the stepper motor, ground switch and servo motor. The stepper motor is used in a 4 pole configuration, although the hardware supports 6 poles. The AIOSL has a built-in stepper motor drive (DRV8813) for directional control [9], and a DAC (MCP4922) for control of the current, hence the motor torque. The stepper motor in the AIOSL is connected with a one-by-four female wire head, which makes it easy to switch it out with the wires coming from the elevator model. To connect the servo and the ground switch, the extra pinheads which are implemented in the AIOSL are used. There are two two-by-five pin headers, each with a 5 volt pin and a ground pin. According to [9], the pin D12 is connected to the ground switch and D15 to the servo. The wire connection between the servo motor and the elevator model is pulled through the printed skirt, up through drilled holes on the second and third floor, and finally through a hole in the back of the cabin.

B. Mechanical design

The model's availability is a crucial consideration. Components and production methods that are readily available are used for components and for the prototyping process. The floors are made in acrylic sheets, which are laser cut in the right shape. The elevator cabin components including doors, the bottom skirt and a mounting spacer for the stepper motor are 3D-printed. All metal parts are easily available off the shelf in hobby shops and some components in hardware stores.

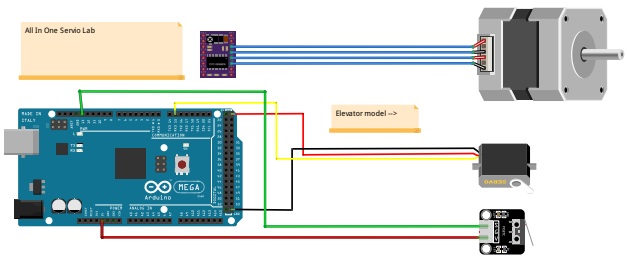

Fig. 3: Connection from the AIOSL to the elevator model.

TABLE III: Mechanical components for the elevator model.

Article	Material	Size	Qty.	Comment
Threaded rod	Steel	Ø 8mm - 470 mm	1	
Lead nut	Brass	Hole 8 mm	1	
V-slot Rat Rig profile	Aluminium	20x20x500 mm	1	
Threaded rod	Steel	M5 - 500 mm	4	
Nut	Steel	M5	8	
Flex axle	Steel	3mm - 8 mm	1	
Lock nut	Steel	5 mm	4	
Tube spacer	PLA	70 mm	24	3D-printed
Short tube spacer	PLA	25 mm	4	3D-printed
Stepper spacer	PLA	42x42x18 mm	1	3D-printed
Bottom skirt	PLA	152x102x68 mm	1	3D-printed
Cabin + cabindoor setup	PLA		1	3D-printed
M3 machine screw	Steel		4	
Bottom floor	Acrylic	150x100x4 mm	1	Laser cut
Mid floor	Acrylic	150x100x4 mm	6	Laser cut
Top floor	Acrylic	150x100x4 mm	1	Laser cut
Door servo			1	
Stepper motor			1	17HS4401
Wires				

In Table III, all the mechanical components needed for setting up the elevator model and integrating it with the AIOSL are described. An exploded view of the elevator model is shown in Figure 4. The threaded rod with a diameter of 8 mm is the type used for controlling the z axis of 3D printers, and is usually available as spare part in most shops selling 3D-printers. The M5 is a standardised threaded rod, and four of them are used for holding the assembly together. The V-slot is a standard RatRig profile. This part may be switched with most profile rods. Small adjustments of the profile on the elevator cabin to fit the selected profile would be necessary.

The main tower part of the elevator model is assembled by

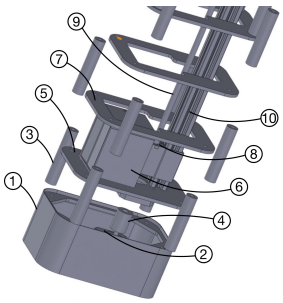

Fig. 4: Exploded view of the elevator model: (1) Bottom skirt, (2) Stepper motor, (3) Tube spacer, (4) Flex axle, (5) Bottom floor, (6) Cabin + Cabin door setup, (7) Mid floor, (8) Lead nut, (9) Threaded rod, (10) V-slot RatRig profile.

first mounting the lock nuts at the end of four M5 threaded rods, and placed in the corner holes from the underside of the stand. The bottom floor plate is then threaded on the rods and placed down in the fitted area in the stand. The stepper is then mounted from the underside with the M3 bolts through the bottom plate and the stepper spacer. This is to make sure that the shaft has the correct height above the plate for the flex axle to be mounted. The ground switch is mounted on the side of the spacer with two screws entered through pre-made holes, making sure that the switch is triggered by the elevator at the bottom floor. Finally, the v-profile is attached to the bottom plate with a 5 mm screw from the bottom in a crescent shaped trail on the stepper spacer. The rest of the floors are then assembled by putting a spacer on each rod and a mid floor section. This is repeated six times creating all the floors. The assembled elevator cabin is then slipped on to the profile and screwed down into the lead nut. Finally, the short spacers are mounted on the rods followed by the top plate, which is secured with two nuts on each rod.

The elevator cabin consists of a main body, two doors and a top section, containing the door mechanism and a servo. The door mechanism is operated by a servo pulling a fishing line through guiding poles on the doors, which again are held up by two rubber bands. The assembly is mounted to the cabin body with four M2 machine screws. The servo is connected by a servo wire through a hole at the bottom two mid floors and then the bottom plate.

C. Software design

In Figure 5 the UML Class Diagram of the elevator model is depicted. The "DAC" class is specialised for using the DAC unit, which controls the current connected to the stepper driver. This is necessary because of the old type of stepper driver implemented in the AIOSL. Similar explanation applies for the "Jmstepper" class too, as this is meant for a driver where the coils are activated separately. The remaining classes are created to accommodate the remaining specifications. The "Elevatordoor" class steers the door servo and keeps track of its current state, simultaneously as it stores the set boundaries for the door. The "Switch" class is used to set up multiple switch inputs as elevator cabin buttons, as well as the floor buttons on the outside. These two are passed along to the "Floorchooser" class, which together with the "Potensiometer" class, passes the selected floor for the queue if up or down buttons are pushed. The "Elevator" class is connected to both the "Elevatordoor" and "Jmstepper" class, as well as the "LCD" class. As the "LCD" class is used in both "Elevator" and "Floorchooser" the LCD object has to be created in the main control process and provided for both classes. The "Queue" class is a queuing operator.

VI. Conclusions and Future Work

This study introduced a comprehensive syllabus of the motion control module for the engineering mechatronics curriculum. The module blends a series of organised theoretical

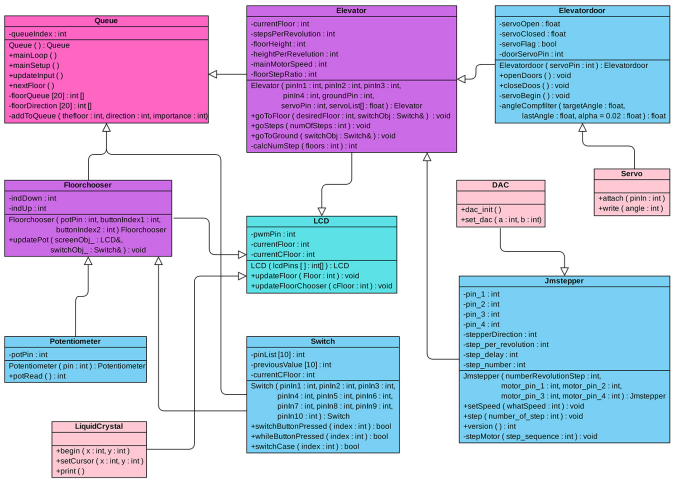

Fig. 5: UML Class Diagram for the elevator model.

sessions with practical and highly engaging laboratory exercises. The course culminates with a group project that focuses on the implementation of an elevator model. The pedagogical effectiveness of the proposed prototype is evaluated based on the students' feedback. In the future, students' feedback will be collected to improve their learning experience and the quality of the provided teaching offer.

Appendix A
Elevator Open Source Repository

The elevator open source repository is available on-line at https://github.com/Microttus/Elevator-model/.

Acknowledgment

This work was supported by the Top Research Centre Mechatronics (TRCM), University of Agder (UiA), Norway.

References

[1] J. Ma, X. Li, and K. K. Tan, *Advanced Optimization for Motion Control Systems*. CRC Press, 2020.

[2] F. Sanfilippo, O. L. Osen, and S. Alaliyat, "Recycling a discarded robotic arm for automation engineering education." in *ECMS*, 2014, pp. 81–86.

[3] 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.

[4] F. Sanfilippo and K. Austreng, "Sustainable approach to teaching embedded systems with hands-on project-based visible learning," *INTERNATIONAL JOURNAL OF ENGINEERING EDUCATION*, vol. 37, no. 3, pp. 814–829, 2021.

[5] University of agder (UiA), "Motion control," https://www.uia.no/en/studieplaner/topic/MAS246-G, 2022, [Online; accessed 07-March-2022].

[6] University of agder (UiA), "Bachelor's programme in mechatronics," https://www.uia.no/en/studieplaner/programme/INGMASK3, 2022, [Online; accessed 07-April-2022].

[7] Arduino, "Arduino," https://www.arduino.cc/, 2022, [Online; accessed 07-April-2022].

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

[9] A. Hørthe, H. Nødland, and B. Preus-Olsen, *All in one Servo Lab, Arduino^{TM}-based laboratory platform for microcontroller operated servo systems*. University of Agder (UiA), 2019.

[10] T. S. Hoang, "An elevator system – requirements document," https://eprints.soton.ac.uk/422715/2/elevator_requirements.pdf, 2022, [Online; accessed 07-April-2022].