Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 52-9 (2019) 42–47

Flexy²: A Portable Laboratory Device for Control Engineering Education

Martin Kalúz [*,**] **Martin Klaučo** [*,**] **Ľuboš Čirka** [*] **Miroslav Fikar** [*]

[*] *Institute of Information Engineering, Automation and Mathematics STU in Bratislava, Radlinského 9, 812 37 Bratislava, Slovakia (e-mail: {martin.kaluz, martin.klauco, lubos.cirka, miroslav.fikar}@stuba.sk)*

[**] *Optimal Control Labs Ltd. Novohorská 42, Bratislava, Slovakia (web: www.ocl.sk, e-mail: info@ocl.sk)*

Abstract: This paper describes a use of laboratory device Flexy^2 for purposes of control education. The device is a simple air flow dynamical system where a computer fan is used as an actuator and a flexible resistor as a sensor. Flexy^2 is designed to directly support practical learning in courses focused on automatic control and programming. The device's properties, principle of operation, dynamical behavior, interfacing features, and general paradigms of usage in education are described. We demonstrate the versatility of the device by providing two case studies that actually take place in our courses. The first is system identification and PID control design via MATLAB and Simulink, and the second is an algorithmic implementation of real time control system using an embedded controller.

Keywords: Laboratory devices, Control education, System identification, PID control, Embedded systems

1. INTRODUCTION

Control engineering education is undergoing a technological revolution in a similar way as its applications in industrial practice. This trend needs to be supported mostly by academia educators. Students, as the future professionals in their field, must be provided with the highest possible education standards, focused not only on their theoretical knowledge, but also technical skills. To acquire such skills, students need to work with a real laboratory equipment, ideally as often as possible. It has been shown in numerous works that hands-on labs using practical experiments increase students' interest and engagement in courses (Cielniak et al., 2013; Lampón et al., 2016).

Many authors show an initiative to use various educational tools in teaching of engineering courses. These cover several fields, such as process control and automation, robotics, and electronics. Huba et al. (2014) employs a thermal-optical device to teach students to design and test controllers for processes of heat transfer. Similarly, an experimental setup for intelligent temperature measurement is shown in Purcaru et al. (2017). Fabregas et al. (2017) demonstrate a remote laboratory with ball and plate system where students can perform various control tasks of trajectory tracking. An educational plant of coupled tanks is used as a remotely controlled experiment by Chacón et al. (2014). A low-cost open hardware approach to robotics education is described in Soriano et al. (2014). Similarly, Catalbas and Uyanik (2017) show a simple yet effective education kit with DC motor for a feedback control systems course. A portable multi-vehicle robotic platform for motion planning algorithms is demonstrated by Yu et al. (2017). Lee et al. (2017) propose a rapid control prototyping system applied to vehicle control.

Most of the mentioned works build their educational equipment around a microcontroller platforms such as Arduino boards. This fact isn't surprising, since the microcontroller-based electronic development platforms are usually inexpensive, feature rich, and easy to use even for people with mediocre electronics and programming knowledge.

At the Institute of Information Engineering, Automation and Mathematics, we have been enriching our courses by a use of real educational devices for some time. We developed 2-axis plotter to support teaching in *Optimization* course (Oravec et al., 2016). This device is used to procedurally visualize basic optimum seeking methods taught in the course. Another device built at our institute is an open-source inverted pendulum (Bakaráč et al., 2017). The most utilized control education device in our courses is Flexy

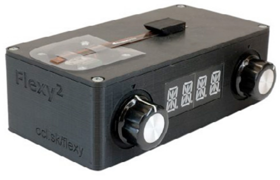
[⋆] The authors gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grants 1/0112/16 and 1/0004/17, the Slovak Research and Development Agency under the project APVV 15-0007. The first author is supported by postdoctoral STU grant. M. Klaučo would like to thank for the financial contribution from the STU in Bratislava Grant Scheme for Excellent Research Teams.

(Kalúz et al., 2018). This device was produced in large quantities and different modifications to support several bachelor and master courses and is frequently used not only in projects of our students but also elsewhere (Demenkov, 2018). Flexy was developed in early 2017 and since, it have been incorporated into courses *Process Control*, *Theory of Automatic Control*, *Identification*, *Technical Means of Automation*, and *Fundamentals of Embedded System*.

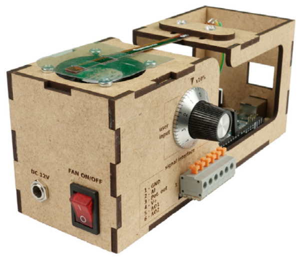
The purpose of this paper is to demonstrate a new version of the device – Flexy^2. We show its main features along with the educational case studies focused on system identification, PID control design, and usage of device in teaching of embedded control.

2. DESCRIPTION OF THE DEVICE

Flexy^2 (Fig. 1a) is a simple dynamical system with one actuator and one sensor. The actuator is a continuously controlled computer fan that propels an air column in an upward vertical direction. The effect of air flow is measured by a flexible sensor that is deployed in the air column. The sensor changes its electrical resistance based on a degree of bend that is caused by hydrodynamic push of air.



(a) Flexy^2 device



(b) Old Flexy device

Fig. 1. Flexy^2 device and its predecessor

The device as a dynamical system exhibits several educationally interesting properties that make control tasks more challenging for students. Firstly, the dynamics is nonlinear. With increased air flow, the flexible sensor is gradually bent and pushed out of the air column, decreasing the sensitivity of the system to further fan speed changes. Secondly, the higher flow rate introduces minor oscillations of the sensor. This effect along with the common noise makes the device ideal for teaching signal processing techniques such as smoothing a filtering.

Flexy^2 is partially based on a design of original Flexy device (Fig. 1b), introducing many improvements, mostly based on usage feedback from students and educators. New device is made of rigid plastic casing that is manufactured by 3D printing process. Second tuning knob has been added to the device (contrary to original Flexy), allowing students to either tune two parameters at same time or to have second custom user input at their disposal.

A small form factor (9 cm width/15 cm length/5 cm height) allows teachers to use Flexy^2 as a portable device and take it to standard computer laboratories where courses are held. The laboratory exercises in the bachelor course *Process Control* are attended by 250 students in average. Since the students are divided in the groups of 12 and maximum two classes are schedule at the same time, we are comfortably able to cover practical experiments for the whole course, using just 24 devices, and without any logistical problems.

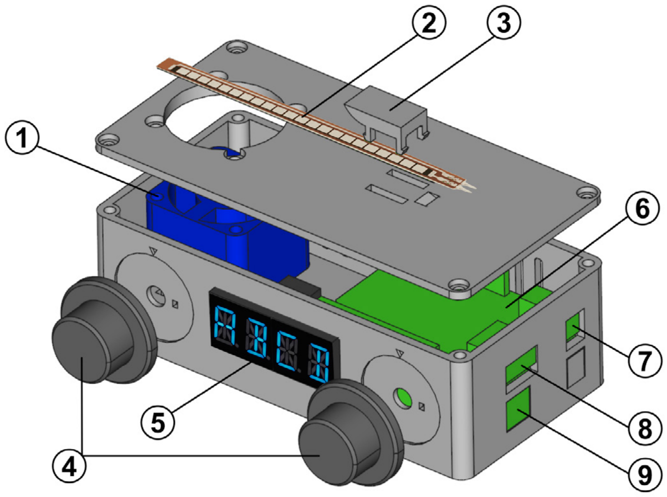


Fig. 2. Individual parts of Flexy^2 device

Flexy^2 device contains multiple functional and supplementary parts (Fig. 2), which are as follows.

1. *Fan*: Device uses a 40 mm computer fan for generating an air column that flows up in a vertical direction and lifts a bendable sensor. This small form factor fan is powered by 12 V DC, consumes up to 4 W of power, and in its highest speed of 11000 RPM it is able to deliver an air flow of 32 m^3 h^{-1}. The speed of fan is controlled via a high frequency switching power circuit using a pulse width modulation technique.
2. *Bendable sensor*: Flexy^2 uses a flexible resistor to measure the effect of air flow. The sensor acts as a variable resistor and its electrical resistance changes with the degree of physical bend. Sensor is connected to a simple signal-conditioning circuitry with a voltage divider and op-amp buffer.
3. *Sensor holder*: The sensor is attached to the device via holder that allows to adjust the overlap of sensor and air column to increase/decrease sensitivity of the process.
4. *Knobs*: Device provides two additional analog user inputs in a form of two potentiometers with knobs.
5. *LCD display*: Quad-alphanumeric LCD display provides a user with the current values of measurements.
6. *Control board*: Device is operated by embedded micro-controller board Arduino UNO Rev3 equipped with custom-made extension board specifically designed for Flexy^2.
7. *DC power connector*: Flexy^2 uses 12V power supply adapter with standardized 5.5/2.1 mm DC barrel power jack.
8. *External signal connector*: 6-pin IDC connector allows user to either attach an additional analog sensor or to interconnect the device with external controller such as PLC.

(9) *USB connector*: A communication between the device and user's computer is carried out via serial-over-USB connection.

3. SOFTWARE AND USAGE

User can interact with the device, to monitor and control it, using a standard computer. The connection to device is provided via USB interface. From the education point of view, Flexy^2 supports two different usage paradigms:

(1) Device is programmed directly on a level of embedded microcontroller. In this case, a whole operating program is written by user in C language, compiled, and uploaded into device, where it runs in a standalone fashion. This paradigm usually takes place in courses focused on programming and embedded control (see Section 5). In these courses, students learn to program low-level functions such as configuring digital and analog IO, reading sensors, processing signals, using digital communication protocols like SPI, I^{2}C and UART, controlling power actuators via PWM, and working with registers and timers. The mentioned functions are mainly dedicated to general setup of the device and its operating condition. On a top of that, students learn how to write algorithms which are responsible for control of the device as a dynamical process. Here, student learn to implement various types of discrete controllers. In the course *Fundamentals of Embedded Systems Control* students are taught to derive discrete PID controllers in multiple forms and to implement them into microcontroller of Flexy^2 as a real-time control system.

(2) Flexy^2 is preprogrammed with a firmware that allows students to use computer as a control device. In this scenario, Flexy^2 provides a control and data acquisition interface over a USB cable. This setup is used in the most of courses dedicated to control design (see Section 4), since it removes the necessity to configure device in any way. Students are provided with the corresponding control interfaces for MATLAB and Simulink to carry out tasks and assignments during laboratory exercises. The topics that are covered in these courses include system identification, PID control, and optimal state feedback control for both continuous-time and discrete-time systems. Moreover, this usage paradigm, in which Flexy^2 provides universal control interface over USB, allows the integration of the device into a teaching process in various computer programming courses. We already provide a Python library that allows students to use Flexy^2 as an interactive tool to test their algorithms in data acquisition and processing.

In the case of MATLAB/Simulink implementation of control algorithms, the controllers are executed directly on a host computer in soft real-time using either, a dedicated timing block if students work with Simulink, or a timer-based function if the control task is implemented in a plain MATLAB code.

4. FLEXY^2 IN TEACHING OF SYSTEM IDENTIFICATION AND PROCESS CONTROL

The identification from step response is one of the most popular and used industrial techniques to obtain dynamical model of a plant. A typical assignment for students in the *Process Control* course is to design a PID controller for a given plant, whose dynamic behavior is not known at a time. One of the approaches to obtain a model of such a plant, which is necessary for analytical PID design methods, is to acquire a step response of the process in some desired operating range. Students will define a series of steps in control signal of Flexy^2 (fan speed) and measure corresponding responses (changes in bend of flex sensor). To obtain a normalized step response, students need to divide every response signal by a magnitude of an excitation signal, shift all responses to the origin, and average them (Fig. 3).

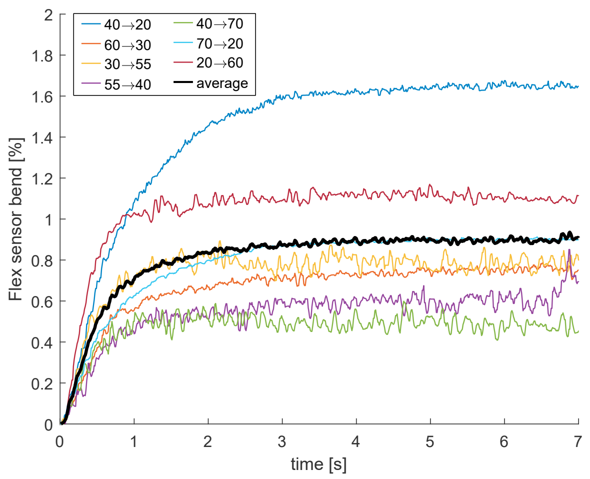


Fig. 3. A set of step responses of flex sensor bend to changes in fan speed. Step responses were obtained for different step sizes and initial conditions. Black line represents an average step response of the process.

Student are taught to use a visual observation to determine whether the step response exhibits a dynamics of first or higher order system. In the case of Flexy^2 in its common configuration, the main portion of dynamics is introduced by a fan (motor) that acts as a first order process. To make the task for students even more challenging, teacher can manually program additional known series dynamics into the device that acts as a filter and increases the order of process model. This will make the device a black box of a sort. To obtain a continuous-time mathematical model in a form of a transfer function, students utilize either an experimental method of system identification from step response or use MATLAB's Plant Identification Tool that is a part of PID Tuner. Resulting mathematical model for Flexy^2 process is

$$G(s) = \frac{0.89}{0.66s + 1}. \qquad (1)$$

The graphical comparison of the process approximation by a first order model (1) is show in Fig. 4.

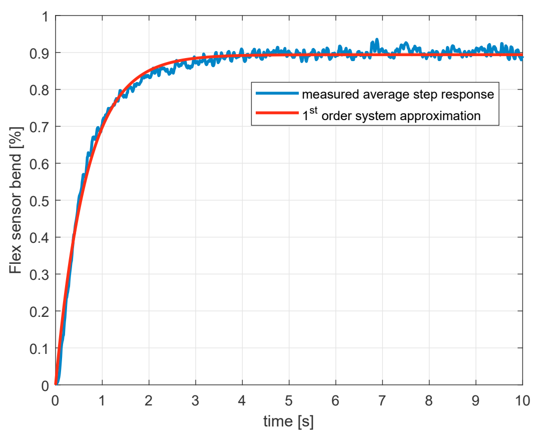


Fig. 4. Comparison of average step response of the process and step response of the model.

In the *Process Control* course students are taught basics of PID control. The main learning objective of this course to understand closed-loop properties and impact of a controller's parameters on control performance. Students use an ideal form of PID and learn how to design its parameters to fulfill basic qualitative measures such as maximum overshoot, settling time, and cancellation of control error.

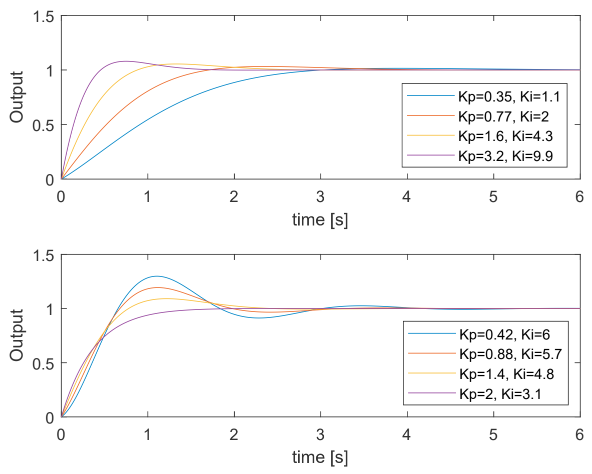


Fig. 5. Various controller performances according to the tuning of response time (loop bandwidth) and transient behavior (phase margin). Top figure shows a dynamical behavior of the closed-loop system according to increasing response time. Bottom figure shows the responses for different transient behaviors, where first two controllers are tuned in a favor of better disturbance rejection and latter two for better reference tracking.

Firstly, students design controllers using MATLAB and test them in a simulation on the previously acquired model of the system. PID parameter selection is usually based on analytical methods, such as Naslin's method to achieve maximum overshoot criterion, or pole placement method where students enforce the closed-loop behavior by directly selecting poles. Students also use experimental methods, namely Ziegler-Nichlos method and Strejc method. Moreover, one of the best practices to help students understand an impact of parameter selection and tuning is a use of an interactive tool. In our case, students use MATLAB's PID Tuner that visualizes response of control loop and allows to manually tune its performance in the means of response time and transient behavior using two sliders. By manipulating the first slider of PID Tuner (response time), student makes a closed-loop response either slower or faster (Fig. 5 - top). By adjusting second slider (transient behavior), student enforce control loop to be more aggressive at disturbance rejection or more robust to process uncertainties therefore more suitable for reference tracking (Fig. 5 - bottom).

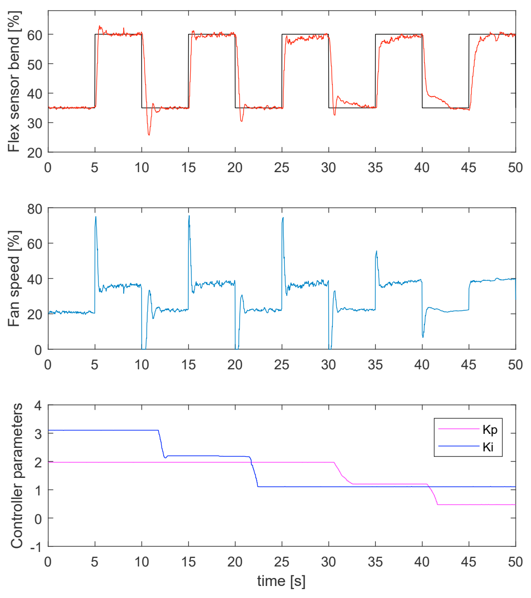


Fig. 6. Control trajectories for step changes of the reference, where user manually decreases values of K_p and K_i in order to observe impact of tuning on control loop performance.

After the controller is designed and tuned, it is implemented in Simulink and connected to block that interfaces a Flexy^2 device with a computer. Students are encouraged to further tune the controller's parameters using two knobs attached to device. These two user inputs are also linked into the Simulink scheme and allow students to increase or decrease values of proportional gain K_p and integral gain K_i. By these means, students can interact with the controller during its execution and learn how values of K_p and K_i influence control trajectories. Figure 6 shows control trajectories of Flexy^2 process for decreasing values of PI controller. Student can see that minor decrease in K_i did not have a significant impact on reference tracking except for slightly reduced overshoot. Lowering a K_p visibly made the controller less aggressive, avoiding physical constraints on input, but as a trade-off, also causing a slower response time. If a student investigates an effect of increasing gains of the controller (Fig. 7), it is obvious that system starts to oscillate as a result of more aggressive control inputs. Moreover, both figures show that process exhibits asymmetric dynamics depending on a direction of control.

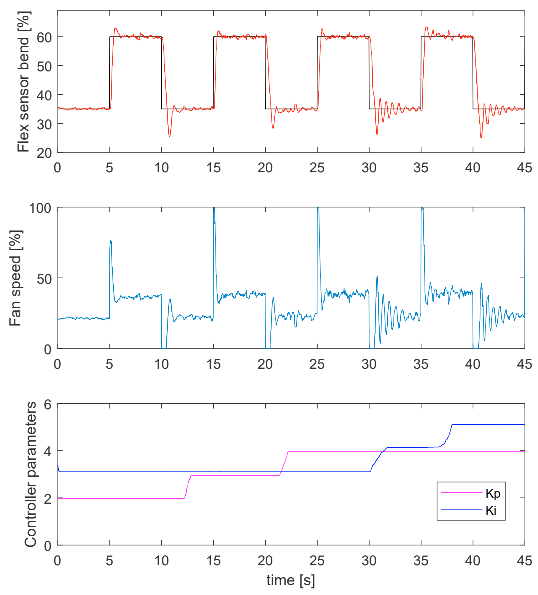


Fig. 7. Control trajectories for step changes of the reference, where user manually increases values of K_p and K_i in order to observe impact of tuning on control loop performance.

5. FLEXY^2 IN TEACHING OF EMBEDDED CONTROL

In this section, we show how students implement a simple discrete-time PID controller directly on a low-level microcontroller Arduino UNO inside a Flexy^2 device. This task takes a place in the course *Fundamentals of Embedded Systems Control*. Unlike the high-level control design languages and programs such as MATLAB/Simulink, LabView and others, microcontrollers do not provide automatic handling of real-time control execution. All the prerequisites for algorithm timing must be programmed by a user.

To ensure that PID controller is evaluated in hard real time, students learn how to utilize microcontroller's timers and their corresponding control registers. A main information source for this purpose is a datasheet of Arduino UNO's microcontroller chip ATmega328P (Microchip Technology Inc, 2018). This source reveals that microcontroller provides three general-purpose timers, two of which utilize 8-bit and one 16-bit counter register. Other control registers of ATmega328P are dedicated to setting up the counting rate and modes in which the timers operate.

The starting point of real time handling is a selection of a timer, which is based on its resolution. A convenient way to implement exact timing is to use a *Clear Timer on Compare Match* (CTC) mode that triggers a program interrupt every time a specific value has accumulated in a timer counter register TCNTx (x is a numerical designator of a timer – 0, 1 or 2). The value of TCNTx is compared to value stored in 16-bit *Output Compare Register* OCRxA/B. There are two such registers in ATmega328P, denoted by letters A and B. The frequency of counting that fills TCNTx is basically a frequency of main clock (16MHz) divided by a value of prescaler, which can be either 1, 8, 64, 256 or 1024. To properly set-up a CTC mode, students need to calculate the desired sampling period T_s of control execution using a knowledge of dynamic behavior of the process. If we assume that 50 Hz is an appropriate sampling frequency ($T_s = 20$ ms), a basic formula to calculate a comparator value OCRxA is

$$\text{OCRxA} = \left\lfloor \frac{f_{\text{CLK}}}{p f_{\text{S}}} - 1 \right\rfloor, \tag{2}$$

where $f_{CLK} = 16 \times 10^6$ Hz is a main clock frequency, f_S is the desired sampling frequency, and p is a prescaler. The comparator value must satisfy the condition

$$\text{OCRxA} \leq 2^N - 1, \tag{3}$$

where N is a timer resolution in bits. Students learn that not every combination of timer resolution N and value of prescaler p will give a feasible solution. If we calculate an OCRxA value for the desired sampling frequency $f_S = 50$ Hz, even the highest possible value of prescaler $p = 1024$ will give a comparator value 312. This value is higher than a maximum value $(2^8 - 1 = 255)$ that can be stored in 8-bit timer counter register. The main message given to students by this result is that 8-bit timer, even in its slowest configuration, counts too fast to achieve 50 Hz sampling frequency and therefore 16-bit timer must be used instead. All prescaler setting for 16-bit timer TCNT1 along with frequency errors caused by quantization are shown in Table 1.

Table 1. Setting of 16-bit timer for sampling frequency $f_S = 50$ Hz

p	OCR1A	f_S	Δf_S
1	319999	50.0000	0.0000
8	39999	50.0000	0.0000
64	4999	50.0000	0.0000
256	1249	50.0000	0.0000
1024	312	49.9201	-0.0799

Since the OCR1A value for $p = 1$ does not meet the condition (3) and $p = 1024$ produces an offset from f_S, student should chose $p \in \{8, 64, 256\}$. Next step is to programmatically set up all required registers of ATmega328P using following assignments:

(1) `cli()` disables global interrupts
(2) `TCCR1A = 0` and `TCCR1B = 0` clears both control registers for Timer 1,
(3) `TCNT1 = 0` resets counter value to 0,
(4) `TCCR1B |= (1 << WGM12)` enables CTC mode on Timer 1,
(5) `TCCR1B |= (1 << CS11) | (1 << CS10)` sets value of prescaler to 64,
(6) `TIMSK1 |= (1 << OCIE1A)` enables CTC interrupts.
(7) `sei()` enables global interrupts

Last step is to define a *Interrupt Service Routine* (ISR) function that takes for an argument a CTC interrupt vector. The body of function `ISR(TIMER1_COMPA_vect)` gets executed with a precise frequency of 50 Hz. On ISR execution the main program is immediately interrupted, setting up a program flag for controller calculation. The controller algorithm is located in a main loop of program and is executed only when the ISR sets the flag variable to true. After the controller execution is finished, the flag variable is set back to false, and control algorithm waits for the next occurrence of CTC interrupt.

After these steps, student already have a timing-based framework for implementation of real-time control algorithms. The PID controller is firstly designed in a continuous time for the continuous model of the process (1) expanded by a zero order hold using Pade approximation. The controller is then discretized by a simple substitution of derivative by a backward difference and integral by a summation term using a trapezoidal rule. This procedure yields an algebraic PID formula either in a position form (4) or incremental form (5) suitable for embedded control system.

$$\begin{aligned} u(k) = Pe(k) + IT_\mathrm{s} \sum_{i=1}^{k} \frac{e(i)+e(i-1)}{2} \\ +D\frac{e(k)-e(k-1)}{T_\mathrm{s}} \end{aligned} \tag{4}$$

$$\begin{aligned} u(k) = u(k-1) + P(e(k)-e(k-1)) \\ +IT_\mathrm{s}\frac{e(k)+e(k-1)}{2} \\ +D\frac{e(k)-2e(k-1)+e(k-2)}{T_\mathrm{s}} \end{aligned} \tag{5}$$

After either (4) or (5) is programmatically implemented on a microcontroller, students use a serial-over-USB plotter to plot the control trajectories to evaluate controller's performance. These tasks are of a simple nature, which is reasonable for the bachelor's study program. Students attend this course in about same time they firstly learn about basics of PID control. Nevertheless, we still encourage students to implement advanced features that often accompany discrete PID algorithms and increase a likelihood of desired control performance. These are for example a suppression of derivative kick effect in a case of step changes in reference, anti windup schemes for control loops that tent to operate on input boundaries, or bumpless transfer for transitions between manual and automatic control mode.

6. CONCLUSION

In this paper we have discussed a educational approach to control system design using Flexy^2 as a teaching tool for laboratory courses. Flexy^2 provides a convenient way to perform various learning tasks that students perform in courses focused on system identification, automatic control, embedded systems and programming. We have shown that some tasks, such as controller parameters tuning can be done in straightforward and interactive matter.

REFERENCES

Bakaráč, P., Kalúz, M., Čirka, Ľ., 2017. Design and development of a low-cost inverted pendulum for control education. In: Proceedings of the 21st International Conference on Process Control. Slovak Chemical Library, Štrbské Pleso, Slovakia, pp. 398–403.

Catalbas, B., Uyanik, I., 2017. A low-cost laboratory experiment setup for frequency domain analysis for a feedback control systems course. IFAC-PapersOnLine 50 (1), 15704 – 15709, 20th IFAC World Congress.

Chacón, J., Beschi, M., Sánchez, J., Visioli, A., Dormido, S., 2014. An experimental framework to analyze limit cycles generated by event-based sampling. IFAC Proceedings Volumes 47 (3), 9051 – 9056, 19th IFAC World Congress.

Cielniak, G., Bellotto, N., Duckett, T., Feb 2013. Integrating mobile robotics and vision with undergraduate computer science. IEEE Transactions on Education 56 (1), 48–53.

Demenkov, M., 2018. Studying aeroelastic oscillations with tensoresistor and arduino. AIP Conference Proceedings 1959 (1).

Fabregas, E., Dormido-Canto, S., Dormido, S., 2017. Virtual and remote laboratory with the ball and plate system. IFAC-PapersOnLine 50 (1), 9132 – 9137, 20th IFAC World Congress.

Huba, T., Huba, M., Bisták, P., Ťapák, P., 2014. New thermo-optical plants for laboratory experiments. IFAC Proceedings Volumes 47 (3), 9013 – 9018, 19th IFAC World Congress.

Kalúz, M., Čirka, Ľ., Fikar, M., 2018. Flexy: An open-source device for control education. In: 13th APCA International Conference on Automatic Control and Soft Computing. Univesrity of the Azores, Ponta Delgada, Portugal, pp. 37–42.

Lampón, C., Costa-Castelló, R., Dormido, S., Sep. 2016. Hands on laboratory for classical nonlinear control systems: The dead-zone case. In: 2016 IEEE Conference on Control Applications (CCA). pp. 816–820.

Lee, Y. S., Jo, B., Han, S., 2017. A light-weight rapid control prototyping system based on open source hardware. IEEE Access 5, 11118–11130.

Microchip Technology Inc, 2018. megaAVR Data Sheet, ATmega48A/PA/88A/PA/168A/PA/328/P. Online, accessible at `https://www.microchip.com/wwwproducts/en/ATmega328p`.

Oravec, J., Kalúz, M., Bakaráč, P., Bakošová, M., 2016. Improvements of educational process of automation and optimization using 2d plotter. In: Preprints of the 11th IFAC Symposium on Advances in Control Education. Vol. 11. pp. 16–21.

Purcaru, D., Purcaru, A., Rădulescu, V., May 2017. Temperature measurement and control system for engineering education. In: 2017 18th International Carpathian Control Conference (ICCC). pp. 258–262.

Soriano, A., Marín, L., Vallés, M., Valera, A., Albertos, P., 2014. Low cost platform for automatic control education based on open hardware. IFAC Proceedings Volumes 47 (3), 9044 – 9050, 19th IFAC World Congress.

Yu, J., Han, S. D., Tang, W. N., Rus, D., May 2017. A portable, 3d-printing enabled multi-vehicle platform for robotics research and education. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). pp. 1475–1480.