The Journal of Engineering and Exact Sciences – jCEC, Vol. 06 N. 05 (2020)
journal homepage: https://periodicos.ufv.br/ojs/jcec
doi: 10.18540/jcecvl6iss5pp0585-0593
OPEN ACCESS – ISSN: 2527-1075

ARDUINO BASED PLATFORM FOR PROCESS CONTROL LEARNING

PLATAFORMA BASEADA EM ARDUINO PARA APRENDIZADO DE CONTROLE DE PROCESSOS

B. C. M. HENRIQUE[1], L. C. M. HENRIQUE[2] and H. M. HENRIQUE[3,*]

[1] Federal University of Uberlandia, Electrical Engineering Department, Uberlandia, Minas Gerais, Brazil
[2] Federal University of Uberlandia, Mechanical Engineering Department, Uberlandia, Minas Gerais, Brazil
[3] Federal University of Uberlandia, Chemical Engineering Department, Uberlandia, Minas Gerais, Brazil

[*] *Corresponding author. Federal University of Uberlandia, Chemical Engineering Department, Uberlandia, Minas Gerais, Brazil, Phone: +55 34 3230-9540 e-mail: humberto@ufu.br (H. M. Henrique).*

A R T I C L E I N F O

Article history:
Received 2020-11-04
Accepted 2020-12-12
Available online 2020-12-12

p a l a v r a s - c h a v e
Real-time control
Arduino applications
Processing applications
PID with Arduino
Process control education

k e y w o r d s
Controlee m Tempo Real
Aplicações em Arduino
Aplicações em Processing
PID com Arduino
Aprendizado de Controle de Processos

A B S T R A C T

This work deals with implementation of an experimental flowrate control unit using free and low-cost hardware and software. The open-source software Processing was used to develop the source codes and user graphical interface and the open-source electronic prototyping platform Arduino was used to acquire data from an experimental unit. Work presents descriptions of the experimental setup, the real-time PID controllers used and theoretical/conceptual issues of Arduino. PID controllers based on internal model control, minimization of the integral of time-weighted absolute error, Ziegler-Nichols, and others were tuned for setpoint and load changes and real-time runs were carried out in order to make real-time use of control theory learned in academy. Results showed the developed platform proved to be suitable for use in experimental setups allowing users compare their ideas and expectations with the experimental evidence in a real and low-cost fashion. In addition, the instrumentation is simple to configure with acceptable level noise and particularly useful for control/automation learning with educational purposes.

R E S U M O

Este trabalho trata da implementação de uma unidade experimental de controle de vazão utilizando hardware e software gratuitos e de baixo custo. O software livre Processing foi usado para desenvolver os códigos fonte e a interface gráfica do usuário e a plataforma de prototipagem eletrônica de código aberto Arduino foi usada para adquirir os dados de uma unidade experimental. O trabalho apresenta descrições da configuração experimental, dos controladores PID de tempo real usados e das questões teóricas/conceituais do Arduino. Os controladores PID com base nos métodos controle por modelo interno, minimização da integral do erro absoluto ponderado no tempo, Ziegler-Nichols e outros foram ajustados para mudanças de setpoint e de carga e execuções em tempo real foram realizadas a fim de fazer uso da teoria de controle em tempo real ensinada na academia. Os resultados mostraram que a plataforma desenvolvida se provou adequada para uso em configurações experimentais, permitindo aos usuários comparar suas ideias e expectativas com as evidências experimentais de forma real e com baixo custo. Além disso, a instrumentação é simples de configurar com nível de ruído aceitável e particularmente útil para aprendizagem de controle automação com fins educacionais.

1. INTRODUCTION

Teaching and research of process control in academy are based extensively on available simulation packages and virtual laboratories. Of course, both have their well-known importance in process control teaching and research. But, unfortunately, experiments with control loops are often limited to these virtual domains and students do not get feedback from physical and real world about impact of control algorithms and their parameters. Reason for this is high hardware and software costs needed to implement control algorithms in real-time fashion. Teaching and research in control and automation in undergraduate and graduate courses are expensive because they involve the use of high-cost proprietary hardware and software. Because of this, control and automation courses are relegated to theoretical approach, leaving students without practical experience of these technologies. This work aims to develop a low-cost, direct, and surprisingly powerful experimental platform for implementing real-time control algorithms. This platform can be used in process control laboratories for teaching and research activities. The platform consists of Arduino boards, a low-cost computer running Windows^{TM} and low-cost Arduino compatible sensors. Sensors and motors are connected to input/output Arduino pins, allowing computer to send and receive signals to/from experimental setups. Arduino pin functions are software programable. Control algorithms were implemented using the open-source Processing software, which allows students to develop their control algorithms and graphical user interfaces with no additional cost. Graphical user interface and communication with Arduino were developed in Processing language.

In the last years, Arduino platforms have gained importance in control and automation applications due to their open-source and low-cost features (Zachariadou et al., 2015; Barber et al., 2013; Granvillano, 2014; Ishikawa and Maruta, 2014; Sobota et al., 2013; Úbeda et al., 2009; Valera et al., 2014). Considering all features of the Processing/Arduino platform as well as results from other researchers, this paper intend to show how worthful proposed platform can be in process control learning.

Remaining content of this paper is organized as follows. Section 2 introduces Arduino and Processing platforms briefly, highlighting their main features. Experimental setup is described in Section 3. Section 4 presents mathematical modelling, model validation of the flowrate dynamical process, experimental results of PID, GMC and MPC controllers intended to control the process flowrate. Section 5 presents results of the impact of the use of proposed platform in academic performance of engineering students in Federal University of Uberlandia. Finally, main conclusions of the work are discussed in Section 6.

2. THE PLATFORM DEVELOPED

Platform developed consists of an Arduino board, a computer, and a software to make all “things” to work. These hardware and software form a perfect match for real-time control applications, combining graphic power of the *Processing* software and the input-output capability of Arduino boards. *Processing* uses Java language and allows to transform Arduino boards into real programmable controllers. In this scenario, students/users can create control algorithms freely using the same tools and workflow concepts that are used in development of industrial control algorithms. Control platform developed specifically consists of an Arduino UNO R3 microcontroller board, an 8th generation Intel Core i7 16GB RAM computer running under Windows^{TM} 10 Pro, Arduino software IDE, *Processing* software IDE, two 12VDC minipumps, a hall-effect flow sensor and a L293d motor shield. Individual components are discussed in the following sections.

2.1 Arduino UNO Board

Arduino is a well-known open-source electronic prototyping platform that consists of a central microcontroller with many built-in features such as digital input/output pins, analog inputs, pulse width modulation (PWM) outputs and others. Arduino boards are inexpensive, flexible, and easy to use for both beginners and professionals, especially for those who would not have access to more sophisticated controllers and more complicated tools. Regarding the software to program Arduino, the same IDE (Integrated Development Environment) is used for all boards and it is available for different OS (Arduino, 2015). This IDE is open and free, as well as easy to get, start and use. C++ language with minor modifications is used as programming language, which enables users create from a simple program based on procedures in a single file to a complex object-oriented program in multiple files. Other relevant aspect of the Arduino platform is its extensive amount of information available, ranging from basic documentation in the official web site to full books for different application fields (Banzi, 2011; Warren, et al., 2011).

Arduino Uno R3 board used in this work is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), an USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller, just connect it to a computer with an USB cable or power it with an AC/DC adapter or battery to get started. In current platform, the board was programmed to act as a slave and its inputs and outputs were used to interact with physical world (Figure 1a). Arduino UNO R3 board was used for acquiring data from a hall-effect flow sensor and sending electrical control signal to two DC minipumps, simultaneously. The aim is to control the flowrate by using minipumps. This board has in its core functions for reading analog signals using a digital analog converter. Arduino UNO board (Figure 1b) has digital input and output pins, as well as analog input and output pins. All six PWM outputs are 8-bit resolution outputs and the analog inputs are 10-bit resolution inputs. The Atmega328's AD converter clock allows acquisition up to 15400 samples per second.

2.2 Arduino IDE

Arduino Integrated Development Environment (IDE) is a multiplatform application written in Java derived from the

Processing and *Wiring* projects. The Arduino IDE is designed to introduce programming for beginners. It includes a code editor for program compiling and loading to boards with a single click. This IDE consists of a simple text editor for writing codes, a message area, a text console, tabs for managing files, a toolbar with buttons for common actions, and several menus. The Arduino IDE also incorporates several libraries. These libraries provide extra functionality for use in sketches and expand capabilities of Arduino boards for manipulating data. A number of libraries come installed with the IDE, but users can also download or create their own.

This capability allows users create many input and output operations in an easy fashion, just defining the two functions following in order to make a functional program:

setup*()* - Inserted at the beginning, which can be used to initialize configuration.

loop() - Call to repeat a command block or wait until it is disconnected.

Arduino IDE uses the GNU toolkit and AVR Libc to compile programs and a command-line program (*avrdude*) to upload programs to boards. *Processing* IDE has also a serial communication library to communicate with Arduino boards in an easy fashion. Therefore, Arduino IDE and the *Processing* IDE communicate with each other through serial communication.

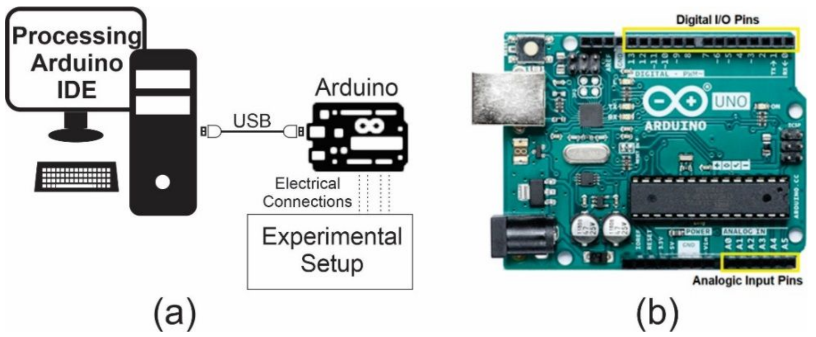


Figure 1 - Arduino/Processing teaching and research platform. (a). Wiring diagram. (b). Arduino UNO R3 details.

2.4 Arduino Shields

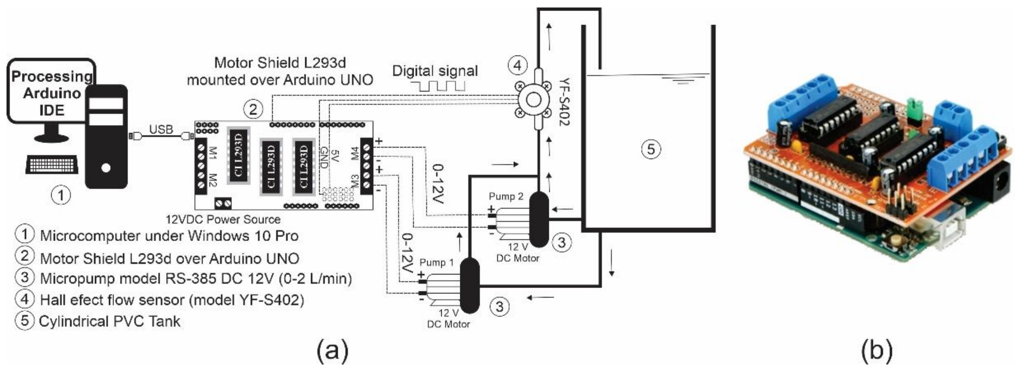
Arduino boards and their clones make use of shields to expand their capability. Shields are printed circuit boards normally attached to top of Arduino boards through a connection powered by pin-connectors. They are expansions that provide several specific functions from engine handling to wireless network systems. In this work the L293D Motor Shield was used to control DC motors. More details about this shield will be given in the following.

2.5 Real-time Controllers

Last step in creating a real-time control platform was the implementation of the control algorithm on the target platform. Several PID control algorithms as presented in Seborg (2011) were implemented in Processing language. This type of controller was chosen because of its successful and vast popularity in industrial applications Seborg (2011). Graphical interface was also implemented in Processing and all inputs, outputs and setpoints were acquired by an Arduino UNO R3. Communication protocol used between master (computer) and slave (Arduino) was the serial standard via an USB cable in asynchronous mode, in which data can be sent and received at any time, with the communication rate parameters (baud rate) of 115200 bps, 8 data bits, no parity and 1 stop bit. This way, it was possible to connect the computer and its devices such as keyboard, mouse and terminals to Arduino and several sensors. The program that implements actions to acquire data from sensors was compiled by Arduino IDE and loaded into the Arduino microcontroller flash memory via USB cable. Program that implements the control strategies was compiled by Processing into a computer running under WindowsTM 10 Pro. Platform allows sampling frequencies up to 15400 samples/s, depending on system dynamic, control algorithm complexity and number of input/output signals acquired.

3. EXPERIMENTAL SETUP

Figure 2 shows a schematic representation of the setup used in this work. Experimental system consists of a small tank from where water is pumped through a hall-effect flowrate sensor (model YF-S402 0-2.0 L/min). Hall-effect sensor is a transducer that under application of a magnetic field responds with a variation in its output voltage. This output voltage is a 5V square wave with variable frequency depending on the measured flowrate. This digital signal is acquired by an Arduino UNO digital channel and the wave period is measured using the *pulseIn*() Arduino IDE function. This sensor was previously calibrated using a weight balance and stopwatch. Two centrifugal minipumps (model RS-385 DC 12V and 0-2 L/min) in parallel were used to pump water through the flow sensor. In this setup, total flowrate is manipulated by changing voltage at pump terminals from 0 to 12 V, using a motor shield for Arduino (model L293D Driver Bridge H) that controls up to 4 DC motors. This motor shield is based on the CI L293D, also known as H bridge, and it can control up to 4 DC motors, 2 servo motors or 2 stepper motors. This shield receives PWM signal from the Arduino board. Two Arduino UNO PWM pins are then used to manipulate the minipump flowrates using the *analogWrite()* Arduino IDE function. Pumped liquid remains in closed circuit, avoiding water waste during operation of the experimental setup.



Processing Arduino IDE
USB
Motor Shield L293d mounted over Arduino UNO
Digital signal
YF-S402
0-12V
Pump 2
Pump 1
12 V DC Motor
12VDC Power Source
① Microcomputer under Windows 10 Pro
② Motor Shield L293d over Arduino UNO
③ Micropump model RS-385 DC 12V (0-2 L/min)
④ Hall efect flow sensor (model YF-S402)
⑤ Cylindrical PVC Tank
(a)
(b)

Figure 2 - Experimental Setup. (a). Schematic diagram of the platform. (b). Details of a DC Motor Shield L293d mounted over Arduino UNO.

4. RESULTS AND DISCUSSION

4.1 System Identification

Process of pumping water through a flow sensor is simple. Based on this, a first order plus dead time (FOPDT) is the first candidate to model the process (Seborg, 2011). Process input is the voltage send to minipumps and the process output is the flowrate measured by flow sensor. Then, the model can be written as:

$$G(s) = \frac{Q'(s)}{U'(s)} = \frac{Ke^{-\theta s}}{(\tau s + 1)} \qquad (1)$$

where q(t) is the process output (flowrate) and u(t) is the process input which is a pulse width modulation (PWM) signal sent to DC motor shield L293d. K, θ and τ are the model parameters. These parameters were estimated using experimental data acquired from an open loop run. PWM pins of the Arduino UNO R3 can generate PWM signal with an 8-bit resolution. As a consequence, a 0-12V signal can be sent to both minipumps by the DC motor shield L293d. This 0-12V was mapped as 0 to 255 bytes (8-bit resolution board, 2^8 bytes) in the Arduino IDE, software that interfaces with the board. In the open loop experiment, an input, u(t), ranging from 90 to 255 bytes was sent from *Processing* to Arduino, from Arduino to motor shield, and from motor shield to both minipumps. Process output (flowrate) was measured by the hall-effect flow sensor. A square wave (digital signal) was generated, and its frequency (Hz) was measured by a digital pin in Arduino board. This frequency was sent to *Processing* and the flowrate was calculated by using a linear equation previously calibrated. A half second sampling time was used. Figure 3 show open loop data. Model time delay (θ) was determined by data inspection and it was fixed as θ = 0.5 s. Other two model parameters of Equation 1 were calculated by nonlinear regression using open loop data, i.e., u(t) versus q(t). Experimental data were zero-mean normalized before estimating parameters for better convergence properties of the optimization package used. Calculated values of the parameters are as follow:

$$G(s) = \frac{Q'(s)}{U'(s)} = \frac{6.69e^{-s/2}}{(0.39s + 1)} \qquad (2)$$

K = 6.69 ± 0.92 mL byte^{-1} min^{-1}, τ = 0.39 ± 0.02 s, and θ = 0.5 s.

Results showed the relative standard deviation of the time constant ($\sigma = 100\sigma_\tau/\tau$ = 5.12%) is smaller than the relative standard deviation of the static gain ($\sigma = 100\sigma_K/K$ = 13.5%). Therefore, the model uncertainty in the static gain is larger than uncertainty in time constant. Figure 3 confirms visually that FOPDT model represents the experimental data reasonably well, with good identification of the process time constant (transient regions), but with some difficulty in steady state regions, especially in regions where flowrate is low.

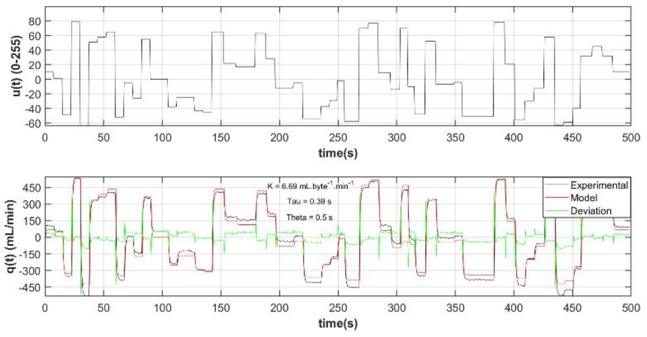


Figure 3 - Open loop run. Experimental data versus mathematical linear model.

4.2 Closed loop operation

Despite model uncertainties, Equation 2 was used to tune PID controllers. Output of this type of controller is calculated based on setpoint deviation (Ogata, 2003; Dazzo and Houpis, 2002). Control action, u(t), calculated by PID controllers takes the form of Equation (3) where parameters K_C, τ_I and τ_D must be tuned:

$$u(t) = K_c\left(e(t) + \frac{1}{\tau_I}\int_0^t e(t)dt + \tau_D\frac{de(t)}{dt}\right) + u_0 \qquad (3)$$

Traditional methods of PID tuning such as internal model control (IMC) by Morari, (1983), Ziegler-Nichols (ZN) by Ziegler and Nichols (1942), minimization of time-weighted absolute error (ITAE) by Lopez and Murril (1967) and Rovira, and Murrill (1969), Åström and Hägglund (AH) by Åström, T. Hägglund (2004), and Cohen and Coon (CC) by Cohen and Coon (1954) were used to tune PID controllers. These methods were chosen because they are classical in most control courses in the world. In order to save space, these methods are not described in this paper. Readers can find more details about them in the literature cited. Control move in Equation 3 was implemented online in Processing IDE and Arduino acquired data from experimental setup in real time. Table 1 show all tuned controllers by using aforementioned methods. Besides of this, two dimensionless performance indexes were used to compare closed loop responses. These indexes are defined as follow:

$$Iu = \frac{1}{(t_\infty - t_0)} \sum_{t=t_0}^{t=t_\infty} t \left| \frac{\Delta u(t)}{u(t)} \right| \tag{4}$$

$$Iy = \frac{1}{(t_\infty - t_0)} \sum_{t=t_0}^{t=t_\infty} t \left| \frac{(q^{sp}(t) - q(t))}{q^{sp}(t)} \right| \tag{5}$$

where $u(t)$, $q(t)$ and $q^{sp}(t)$ are the manipulated variable (control move), controlled variable and setpoint at sampling instant "t", $\Delta u(t)$ is the control move (effort) at same sampling instant, t_0 and t_∞ are the initial and final instant of experimental run. Of course, good control performances yield small "Iu" and "Iy" values.

Table 1 - P, PI and PID parameters tuned by using identified FOPTD model of the real plant.

Method	P	PI	PID
IMC	--	$K_c = 0.0657$ $\tau_I = 0,3920$ $(\tau_C = \tau_I)^{(*)}$	--
ITAE (setpoint)	--	$K_c = 0.0701$ $\tau_I = 0.4783$	$K_c = 0.1173$ $\tau_I = 0.6435$ $\tau_D = 0.1514$
ITAE (Load)	--	$K_c = 0.1012$ $\tau_I = 0.6863$	$K_c = 0.1611$ $\tau_I = 0.5571$ $\tau_D = 0.1903$
Ziegler-Nichols (ZN)	$K_c = 0.1172$	$K_c = 0.1055$ $\tau_I = 1.6650$	$K_c = 0.1406$ $\tau_I = 1.0000$ $\tau_D = 0.2500$
Cohen-Coon (CC)	$K_c = 0.1670$	$K_c = 0.1179$ $\tau_I = 0.4927$	$K_c = 0.1986$ $\tau_I = 0.8544$ $\tau_D = 0.1474$
Åström and Hägglund (AH)	--	$K_c = 0.0537$ $\tau_I = 0.4122$	--

$[K_c]$ in mL^{-1}min.byte, $[\tau_I]$ in seconds and $[\tau_D]$in seconds. $^{(*)}\tau_c$ is the closed-loop time constant (tuning parameter).

4.2.1 Proportional and integral action

Steady-state error concept in control loops is addressed in regular control courses. Students learn proportional action (K_c) is able to reduce rise time, but it is not able to eliminate steady-state error. They also learn only integral action (τ_I) in control loops is able to eliminate steady-state error, but it can make transient response worse. In order to demonstrate differences between these two control modes (P and I) in a real case, the two ZN controllers (P and PI) from Table 1 were implemented in *Processing* and tested for real setpoint tracking. Results from Figure 4 reveal control responses exactly as predicted in control literature, i.e., P controller response is faster than PI controller response ($Iu_P < Iu_{PI}$). However, P controller was not able to eliminate offset. Larger Iy_P value than Iy_{PI} value was caused by the large offsets present in P controller responses. PI controller eliminated offset as predicted by control theory.

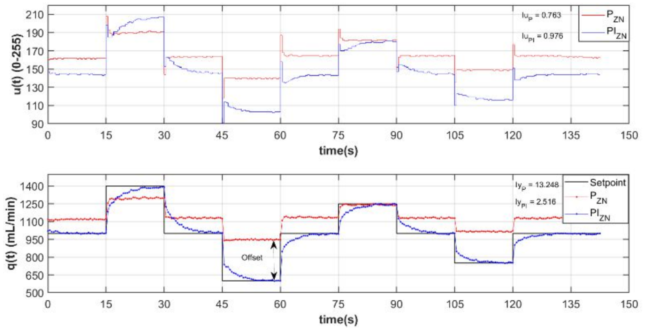


Figure 4 - P and PI responses and control moves for setpoint changes. Controllers tuned by the Ziegler-Nichols method.

4.2.2 Integral and derivative action

Control literature states derivative control (τ_D) has effect of increasing closed loop stability, reducing overshoots, and improving transient response. Results in Figure 5 show control loop behavior of two controllers (PI and PID) in a real case. Both controllers were tuned using ITAE method and they are shown in Table 1. They were implemented in *Processing* and tested for real setpoint tracking. Controlled responses of both controllers were similar ($Iy_{PI} \approx Iy_{PID}$). But in terms of control move, the PI performance was better than PID because $Iu_{PI} < Iu_{PID}$ (Figure 5). In addition, at t = 105 s small oscillations in the PID response can be seen. This response degeneration could be likely caused by noise in the acquired data. Noise is present in all real measurements, and it can degenerate numerical calculation of error derivative in Equation 3. Response degeneration can also be caused by plant/model mismatch. Figure 3 reveals some deficiency of the model to represent the real plant in low flowrate region, where fluctuations have appeared. It well known the derivative action is almost never used in industrial applications (Seborg, 2011) because of noise in real data. Students can learn about this with real data and can face this challenge in a practical situation, for instance, designing digital filters for noise.

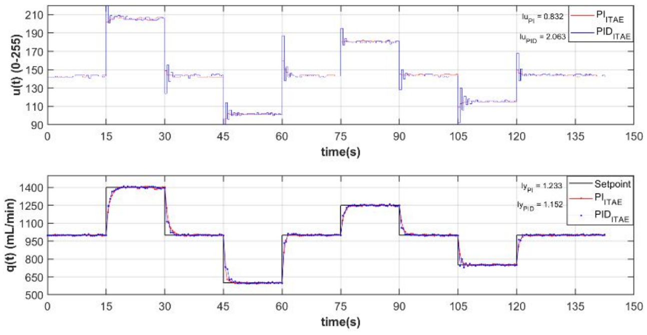


Figure 5 - PI and PID responses and control moves for setpoint changes. Controllers tuned by ITAE method.

4.2.3 Comparison of tuning methods

Other classical controllers were also tested in servo problem in order to verify their real performances. Figure 6 show results for PI controllers tuned using IMC, CC, and AH techniques (Morari, 1983; Cohen and Coon, 1954; Åström and Hägglund, 2004). All controllers carried out the proposed setpoint tracking. PI_{IMC} and PI_{AH} had good and similar performances with similar values of the Iu and Iy indexes. PI_{CC} presented some over/undershoots and oscillations from t = 45 s to t = 60 s in low flowrate operational region. Aggressive response of the PI_{CC} in comparison with the PI_{IMC} and PI_{AH} can be explained by greater K_C and smaller τ_I than PI_{IMC} and PI_{AH} (Table 1) as stated by control literature. This result is supported by Rivera and Morari (1986) that states when $(\theta/\tau) < 2$ the Cohen-Coon method presents bad robustness characteristics. On the other hand, PI_{IMC} and PI_{AH} had good similar responses because their K_C and τ_I are alike (Table 1).

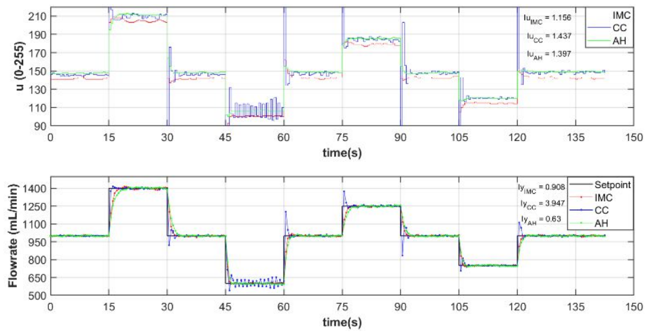


Figure 6 - PI responses and control moves for setpoint changes. Controllers tuned by the IMC, CC, and AH methods.

Several results in control literature (Lopez and Murril, 1967; Rovira et al., 1969) also suggest making distinct tuning for load and setpoint changes. In order to verify this statement in practical situations, the ITAE controllers (Lopez and Murril, 1967; Rovira et al., 1969) presented in Table 1 were tested for setpoint and load changes. Two different PID controllers were tuned based on setpoint ITAE and load ITAE rules. Controller parameters are shown in Table 1. Figure 7 show the results. For setpoint tracking the setpoint ITAE controller has clearly a better performance than load ITAE controller, as control theory states (Lopez and Murril, 1967; Rovira et al., 1969). This better performance is also supported by smaller Iu and Iy indexes (Figure 7) of setpoint ITAE controller. Concepts about stability and oscillatory control loop behavior can be investigated by using this simple experimental setup. In this case, the poles of characteristic equation ($1+ G_cG_{ol} = 0$, [11]) were calculated:

Setpoint ITAE controller:

$$p_1 = -3.81;\ p_2 = -2{,}76\ and\ p_3 = -1{,}70 \tag{6}$$

Load ITAE controller:

$$p_1 = -9.12;\ p_2 = -1{,}68 + 1{,}38i\ and\ p_3 = -1{,}68 - 1{,}38i \tag{7}$$

Reason for oscillations in load ITAE controller behavior can be now explained. This controller has two complex poles with negative real part. Therefore, this control loop response will always be oscillatory with over/undershoots and damped oscillations for step setpoint changes, exactly as seen in Fig. 7. This figure also reveals a worse behavior of this controller at operational region with low PWM. This is likely caused by plant/model mismatch. Figure 3 show clearly a model deficiency in the static gain of the process. Therefore, the plant/model mismatch is more severe in this region of operation and it can be likely responsible for worse performance of controller in this region. On the other hand, expected control loop response of the setpoint ITAE controller will be monotonic with no oscillations and smooth settlement in setpoint as also seen in Figure 7, because its closed loop poles are all negative real numbers.

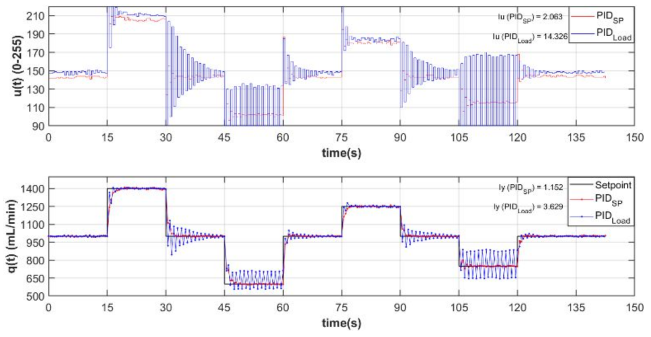


Figure 7 - PID responses and control moves for setpoint changes. Controllers tuned by the Setpoint ITAE and Load ITAE methods and used for setpoint changes.

Inverse situation was also investigated, the use of setpoint and load ITAE controller for disturbance rejection (load problem). For this case, step changes were introduced in PWM signal sent to minipump #2, emulating process disturbances. PWM signal to minipump #1 is only manipulated variable. Figure 8 show the results. Load ITAE controller was able to reject the disturbances faster than setpoint ITAE controller, exactly as predicted by control theory (Lopez and Murril, 1967; Rovira et al., 1969). This result is also supported by its smaller Iy index than setpoint ITAE controller at expenses of more aggressive control moves ($Iu_{PID\ Load} > Iu_{PID\ Setpoint}$, Figure 8). Oscillations noted in load PID response is caused by presence of complex poles at closed loop transfer function as before mentioned.

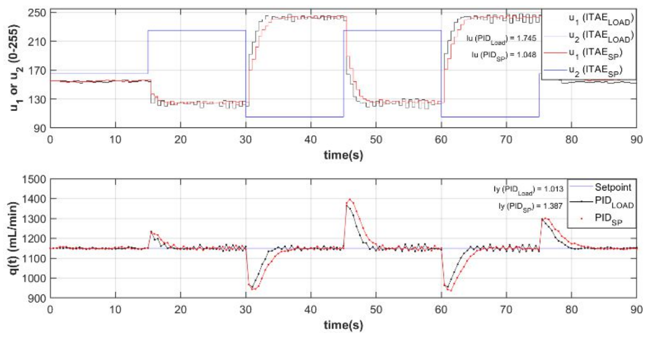


Figure 8 - PID responses and control moves for disturbance rejection. Controllers tuned by the Setpoint ITAE and Load ITAE methods and used for load changes.

4.2.4 Advanced control methods

Advanced process control (APC) refers to a broad range of techniques and technologies implemented within industrial process control systems, such as feedforward, decoupling, and inferential control. APC includes also techniques based on process model, such as Model Predictive Control (MPC) and Generic Model Control (GMC). The flowrate control problem approached can be controlled reasonably by PID controllers as seen before. Despite this control process is quite simple, a MPC and GMC were developed to control the flowrate in a real-time fashion by using the proposed platform. The objective is to show the proposed platform is flexible enough to implement both conventional and advanced control techniques using low-cost devices.

Generic Model Control

Generic Model Control (GMC) is a control method developed by Lee and Sullivan (1988) that directly imbeds a process model in the control algorithm. The process model can be linear or nonlinear. The basic idea this method is to calculate the control actions in order to make the controlled output to follow a reference trajectory q*(t) as shown in Equation 8:

$$\frac{dq^*}{dt} = K_1[q^{sp}(t) - q(t)] + K_2\int_0^t [q^{sp}(t) - q(t)]dt \qquad (8)$$

here $q^{sp}(t)$ is the setpoint for the controlled output $q(t)$, K_1 and K_2 are controller tuning parameters. The controlled output is the flowrate. Equation (1) in time domain can be written as follows:

$$\frac{dq(t)}{dt} = \frac{1}{\tau}[Ku(t) - q(t)] \qquad (9)$$

Control action $u(t)$ is calculated imposing $\dot{q^*}(t) = \dot{q}(t)$. So, substituting Equation into Equation 9 and solving for $u(t)$, the control action is given as follows:

$$u(t) = \frac{1}{K}\left\{q(t) + \tau K_1[y^{sp}(t) - q(t)] + \tau K_2\int_0^t [q^{sp}(t) - q(t)]dt\right\} \qquad (10)$$

Equation 10 was solved online every sampling time in *Processing* and the calculated control action u(t) was sent to experimental plant. In the next sampling instant, all calculations were repeated, and this process continued up to end of the experimental run. Two experimental runs were carried out using the tuned parameters $K_1 = 2.0\ s^{-1}$, $K_2 = 0.25\ s^{-2}$ and $K_1 = 1.0\ s^{-1}$, $K_2 = 0.5\ s^{-2}$. Fig. 9 show the results. Both controllers tracked the setpoint with no offset and with some over/undershoot in setpoint transitions. GMC$_2$ presented oscillatory behavior for setpoint changes in lower values of flowrate. Oscillatory behavior can be addressed re-tuning parameters K_1 and K_2. Results from Fig. 9 show a smaller performance index (Iy) for GMC$_1$ than GMC$_2$.

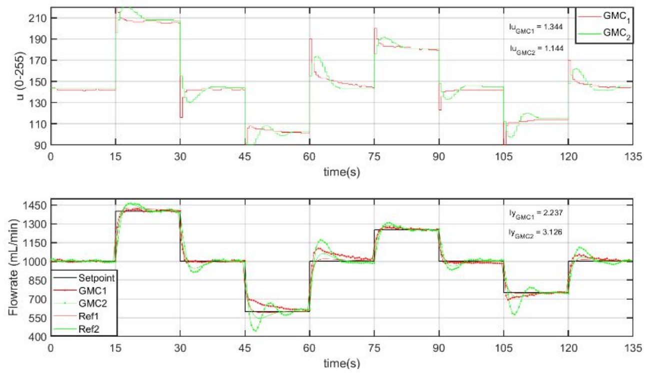


Figure 9 - GMC responses and control moves for setpoint changes. $K_1 = 2.0\ s^{-1}$, $K_2 = 0.25\ s^{-2}$ for GMC$_1$ and $K_1 = 1.0\ s^{-1}$, $K_2 = 0.5\ s^{-2}$.

On the other hand, GMC$_2$ presented smaller Iu index (smaller control moves) than GMC$_1$. These results are a consequence of the tuning parameters K_1 and K_2. Results from Figure 9 also show the controlled output followed the reference trajectory satisfactorily for setpoint changes in higher values of flowrate. But the controlled output deviated from reference trajectory for setpoint changes in lower values of flowrate. This result is likely a consequence of plant/model mismatch. Plant model is less accurate in lower values of the flowrate than higher values of the flowrate, as can be seen in Figure 3.

Model Predictive Control (MPC)

Model predictive control (MPC) is an advanced method of process control that is used to control a process subject to a set of constraints in manipulated and controlled variables. It has been used in process industries, chemical plants, and oil refineries since the 1980s. First-generation MPC systems were developed independently in the 1970s by two pioneering industrial research groups. Dynamic Matrix Control (DMC) developed by Shell Oil (Cutler and Ramaker, 1980) and the approach by Adersa (Richalet et al., 1978) have quite similar capabilities. An adaptive MPC technique, Generalized Predictive Control (GPC), developed by (Clarke and Mohtadi, 1987) has also received considerable attention. Model predictive control has had a major impact on industrial practice. For example, an MPC survey by (Qin and Badgwell, 2003) reported that there were over 4,500 applications worldwide by the end of 1999, primarily in oil refineries and petrochemical plants. In these industries, MPC has become the chosen method for difficult multivariable control problems that include inequality constraints. Basic idea is to use the process model to predict future values of the outputs and calculate the process inputs in order to minimize the distance between setpoints and predicted process outputs. Input changes are calculated based on both predictions and measurements. In addition, process inputs and outputs can be subject to constraints. Figure 10 summarizes the MPC approach.

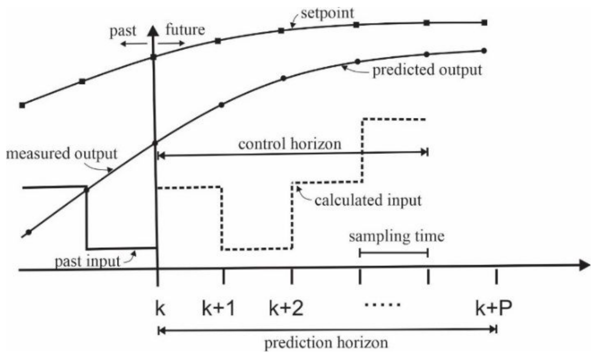


Figure 10 - Model predictive control approach.

MPC is popular in industry due to some important advantages, such as constraints on inputs and outputs that can be considered in a systematic manner, control calculations that can be coordinated with the calculation of optimum setpoints, the approach can be directly extended to multiple-input-multiple-output processes, and its use for complex problems (large time delays and high-order dynamic). In this sense, for the current single input single output system, the MPC can be mathematically transformed into an optimization problem as follow:

$$\min_{(u_1, u_2, \ldots, u_H)} J = \sum_{i=1}^{P}\left(q_i^{sp} - q_i\right)^2 + \sum_{i=1}^{H} w_i(\Delta u_i)^2 \qquad (11)$$

Subject to:

$$\leq u_i \leq u_{max} \qquad (12)$$

$$|\Delta u_i| \leq \Delta u_{max} \qquad (13)$$

where q_i is the controlled variable at *i*th sampling time over the prediction horizon, q_i^{sp} is the setpoint at *i*th predicted sampling time over the prediction horizon, u_i is the manipulated variable at *i*th sampling time over the control horizon, $\Delta u_i = u_i - u_{i-1}$,

w_i is the weighting parameter penalizing large changes in u_i, P is the prediction horizon, and H is the control horizon. Next, the MPC represented by Equation 11 to 13 was applied to the real system studied. Mathematical model in Equation 9 was used into MPC to calculate control moves. The optimization problem (11)-(13) was posed as a successive quadratic problem (SQP) and solved online in *Processing*. Only first control move was sent to experimental plant. In the next sampling instant, new control moves were calculated again, and all calculations were repeated up to end of the experimental run. Three experimental runs were carried out by using prediction horizon equal to 50 sampling instants, control horizon equal to 15 sampling instants, weighting parameter w_i equal to 0.5, and $|\Delta u|_{max} \leq 5$ bytes, $|\Delta u|_{max} \leq 10$ bytes, $|\Delta u|_{max} \leq 20$ bytes. Figure 11 shows the results. All three controllers tracked setpoint reasonably with no constraint violations. Results from Figure 11 show the best performance index (Iy) was of the MPC$_3$, followed by MPC$_2$, and for MPC$_1$, as predicted by MPC control theory. On the other hand, the best Iu index (lower control move) was of MPC$_1$, followed by MPC$_2$, and for MPC$_3$. These results are a consequence of the constraints used ($|\Delta u|_{max}$), i.e., lower $|\Delta u|_{max}$ less aggressive is the MPC, and smoother the system response. It can be noted that the smoothness of controller can be directly addressed in MPC formulation adjusting the value of $|\Delta u|_{max}$. In PID control this question is addressed indirectly by tuning controller's parameter, and this is can be a time-consuming step.

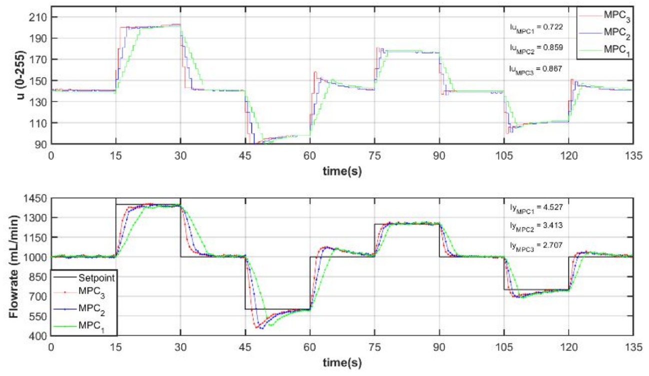


Figure 11 - MPC responses and control moves for setpoint changes. The constraints were |Δu|≤5 bytes for MPC1, |Δu|≤10 bytes for MPC2 and |Δu|≤20 bytes for MPC3.

5. PROCESS CONTROL LEARNING BY USING LOW-COST PLATFORM APPROACH

The process control course at the Federal University of Uberlândia (UFF) is offered to chemical engineering and control and automation engineering students. Course approach has been based exclusively on theoretical aspects of control theory using only simulation packages and virtual laboratories for control practices due to high cost to implement real-time laboratory facilities. Because of this, all experiments with control loops were limited to virtual domains. But in 2015, the control process course approach was changed, and a new methodological approach was implemented, incorporating low-cost and real-time experiments, instead of only virtual ones. Low-cost control experiments such as flowrate, level, pH, and temperature control units based on proposed platform were developed and incorporated in the control process course offered to chemical engineering and control and automation engineering students. Students' performance in the process control course before and after incorporating the new methodological approach was evaluated from years 2000 to 2018. Usually, from 40 to 50 students have attended this course every year. The percentage of students that have passed in process control course during this time was used as performance index, considering only the years supported by same teacher. This is important to eliminate the effect of different teachers on student's performance. This way, the presence or absence of the new methodological approach is the unique variable affecting the students' performance from years 2000 to 2018. Results can be seen in Figure 12.

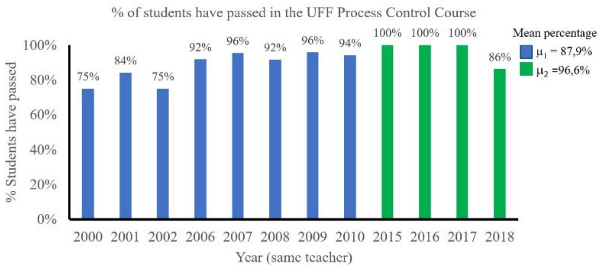


Figure 12 - Students' performance in the process control course of the Federal University of Uberlandia from 2000 to 2018.

The mean percentage of students that have passed in control process course before using the new approach is 87,9% and 96,6% after using it, respectively. In order to have a fair comparison between these two means, statistical inference on both means is performed. We have considered $x_1, x_2, \ldots, x_{n1}$ is a random sample of n_1 observations from population 1 with mean μ_1 and variance σ_1^2 (percentage of students have passed in the process control course before using new approach) and $y_1, y_2, \ldots, y_{n2}$ is a random sample of n_2 observations from population 2 with mean μ_2 and variance σ_2^2 (percentage of students have passed in the process control course after using new approach). Both populations were considered independent and normally distributed. We now consider hypothesis testing on the difference in the means μ_1 and μ_2 of these two normal populations. Thus, the null hypothesis is stated as H_0: ($\mu_2 - \mu_1 = 0$) or $\mu_2 = \mu_1$ and alternative hypothesis as H_1: ($\mu_2 - \mu_1 > 0$) or $\mu_2 > \mu_1$. [26] developed a "z_0" statistic that can be used to test the null and alternative hypotheses (H_0 and H_1). So, the hypothesis testing is summarized as follows:

$$\text{Null hypothesis: } H_0 : \mu_2 = \mu_1 \tag{14}$$

$$\text{Alternative hypothesis: } H_1 : \mu_2 > \mu_1 \tag{15}$$

$$\text{Test statistic: } z_0 = (\mu_2 - \mu_1) \Big/ \sqrt{\sigma_1^2 / n_1 + \sigma_2^2 / n_2} \tag{16}$$

$$\text{Rejection criteria: } z_0 > z_\alpha \tag{17}$$

The statistical significance level of the test is denoted as α and it represents the probability of rejecting the null hypothesis when it is true. Typical value of α = 0.05 (or 5%) was used. Table 2 shows the result of this test. Since $z_0 = 1.824$ is greater than $z_{0.05} = 1.645$, we rejected H_0: $\mu_2 = \mu_1$ at the $\alpha = 0.05$ level and conclude that new methodological approach incorporating low-cost and real-time experiments instead of only virtual ones increased the percentage of students that have passed in control process course.

6. CONCLUSION

This work drove with an open source and low-cost platform based on Arduino and *Processing* software to explore important concepts in control and automation knowledge field. Results showed the proposed platform (hardware plus software) is efficient for use in experimental process control setups. Instrumentation has an easy configuration, with low level of noise and low cost. Therefore, experimental setups like this can be built for didactic and research purposes in a learning environment. Statistical result based on hypothesis test showed proposed platform had a positive impact on the percentage of students have passed in the process control course of the Federal University of Uberlandia in years from 2000 to 2018. This result supports the proposed platform can be effective for process control learning in engineering courses. In addition, it is possible to approach several others important concepts of control literature such as dynamic modelling, system identification, transfer functions, controller tuning, dynamic system stability, in real-time fashion with low-cost investments.

From a pedagogical point of view, the advantage of this teaching and research approach is that the student crosses his ideas and expectations with experimental evidence, gradually becoming competent to deal with different theories of control and automation facing up to experimental evidence acquired in real problems. In conclusion, this teaching and research platform renews the student and teacher roles involved in teaching and research in process control and automation.

REFERENCES

Arduino, Arduino web page, July 2015. Available at: http://www.arduino.cc. Accessed on May 2020.

Åström, K. J.; Hägglund, T. Revisiting the Ziegler-Nichols step response method for PID control, **Journal of Process Control**, n. 14 (6), p. 635-650, 2004.

Banzi, M. Getting Started with Arduino, 2nd Edition. O'Reilly, USA, 2011.

Barber, R.; De La Horra, M.; Crespo, J. Control Practices Using Simulink with Arduino as Low-Cost Hardware. PROC. OF THE 10TH IFAC SYMPOSIUM ON ADVANCES IN CONTROL EDUCATION (ACE2013), Sheffield, UK, August. 2013.

Clarke, D. W.; Mohtadi, C.; Tufts, P. S. Generalized Predictive Control—Part I. The Basic Algorithm, **Automatica**, n. 23, p. 137. 1987.

Cohen, G. H.; Coon, G. A. Theoretical Considerations of Retarded Control, **Trans. ASME**, n. **75**, p. 827. 1953.

Cutler, C. R.; Ramaker, B. L. Dynamic Matrix Control—A Computer Control Algorithm, PROC. JOINT AUTO. CONTROL CONF., Paper WP5-B, San Francisco. 1980.

Dazzo, J. J.; Houpis, C. H. Análise e projeto de sistemas de controle lineares. Guanabara Dois, Rio de Janeiro. 2002.

Granvillano, C. Arduino as a programmable logic controller (PLC). 2014. Available at: http://www.open-electronics.org/arduino-as-aprogrammable-logic-controller-plc/. Accessed on June 2015.

Ishikawa, M.; Maruta, I. Rapid prototyping for control education using Arduino and open-source technologies. PROCEEDINGS OF 8TH IFAC SYMPOSIUM ON ADVANCES IN CONTROL EDUCATION, Kumamoto, Japan, October. 2009.

Lee, P. L.; Sullivan, G. R. Generic Model Control (GMC), **Comput. Chem. Engng**, n. 12 (6), p. 573-580, 1988.

Lopez, A.M.; Murril, P. W. Controller Tuning Relationships Based on Integral Performance Criteria, **Instrumentation Technology**, n. 14 (11), p. 57. 1967.

Montgomery, D. C.; Runger, G. C. Applied Statistics and Probability for Engineer*s*, 7th Edition, John Wiley & Sons. Inc, 2018.

Morari, M. Internal Model Control - Theory and Applications, IFAC PROCEEDINGS VOLUMES, n. 16 (21), p. 1-18. 1983.

Ogata, K. Engenharia de Controle Moderno, 4ª edição, Guanabara Dois, Rio de Janeiro. 2003.

Qin, S. J.; Badgwell, T. A. A Survey of Industrial Model Predictive Control Technology, **Control Eng. Practice**, n. 11, p. 733. 2003.

Richalet, J.; Rault, A.; Testud, J. L.; Papon, J. Model Predictive Heuristic Control: Applications to Industrial Processes, **Automatica**, n. 14, p. 413. 1978.

Rivera, D. E.; Morari, M.; Skogestad, S. Internal model control. 4. PID controller design, **Ind. Eng. Chem. Process Des. Dev.**, n. 25, p. 252-265. 1986.

Rovira, A. A; Murrill, P. W.; Smith, C. L. Tuning Controllers for Setpoint Changes, Technical Report, DTIC Document, 1969.

Seborg, D. E.; Mellichamp, D. A.; Edgar, T. F.; Doyle III, F. J. Process Dynamics and Control, Third Edition, John Wiley & Sons. Inc, 2011.

Sobota, J.; Pišl, R.; Balda, P.; Schlegel, M. Raspberry Pi and Arduino Boards in Control Education (I). PROC. OF THE 10TH IFAC SYMPOSIUM ON ADVANCES IN CONTROL EDUCATION (ACE2013). Sheffield, UK. August. 2013.

Úbeda, D.; Gil, A.; Lucas, J. A.; Jiménez, L. M.; Reinoso, Ó. Ardilla: plataforma de prácticas docentes on-line mediante Arduino. PROC. OF XXX JORNADAS DE AUTOMÁTICA, Valladolid, Spain, September. 2009.

Valera, A.; Soriano, A.; Vallés, M. Low-Cost Platforms for Realization of Mechatronics and Robotics Practical Works. **Revista Iberoamericana de Automática e Informática Industrial RIAI**, n. 11 (4), p. 363-376. 2014.

Warren, J. D.; Adams, J.; Molle, H. Arduino Robotics (Technology in Action). Apress, USA, 2011.

Zachariadou, K.; Iasemides, K. Y.; Trougkakos, N. Experiences on using Arduino for laboratory experiments of Automatic Control and Robotics, **IFAC-Papers OnLine** n. 48-29 , p. 105–110. 2015.

Ziegler, J. G.; Nichols, N. B. Optimum settings for automatic controllers, TRANSACTIONS OF THE ASME. n. 64, p. 759–768. 1942.