Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 55-17 (2022) 296–301

Speed control of a CPU fan ⋆

Ignacio Alvarado Jose A. Borja Richard Haes
David Muñoz de la Peña.

Departamento de Ingeniería de Sistema y Automática, Universidad de Sevilla, Sevilla 41092 Spain (e-mails: ialvarado@us.es, jaborja@us.es, richard9661h@gmail.com, dmunoz@us.es).

Abstract: This work presents a minimalist lab equipment, the speed control of a CPU fan using an Arduino, which, because of the low price of its components, is particularly suitable as a take-home lab for courses implementing a project-based learning methodology. This system presents a multidisciplinary challenge in which students need to apply knowledge of electronics, programming, signal processing, and control. By assembling and controlling this simple device, which consists of only five components, students can learn basic mechatronics and control concepts at the undergraduate level.

Keywords: Education, control, project-based learning, home lab.

1. INTRODUCTION

Any educational branch has as its main goal to equip students with the ability to translate the theoretical knowledge acquired throughout their education, into real-life practical cases. However, this is not a simple task, since it is such a broad and subjective problem, as well as being dependent on the needs of the students and the branch of knowledge, that it would be impossible to define a single method with which to tackle it.

This challenge becomes more complex if it is focused on the teaching of control systems, since, in order to apply a control algorithm in a real system, it is necessary to have an idea of various disciplines that are independent of control theory, such as electronics, signal processing and conditioning, programming, system modeling and identification, use of simulation tools, etc.

Therefore, in the laboratory sessions of control subjects, where the time available is usually very limited, it is typical that students, in order to avoid having to acquire and apply the knowledge of the above disciplines and can focus on the issue at hand, are provided with a simplified and conditioned system, so that the only pending task is to write the control algorithm and analyze the correct operation. The problem with this approach comes when the student has to control a real system: although he knows how to design a controller, he does not know how to implement it.

A popular pedagogy to address this kind of problems is project-based learning (PBL), Perrenet and Smits (2000) Lehmann et al. (2008). This methodology starts by raising a set of questions to start with. As the project evolves, answers to these questions arise, providing the student with the required set of knowledge. Project-based learning engages students in exploring answers to important and meaningful questions through a process of research and collaboration. Students ask questions, make predictions, design solutions, and learn how to collect and analyze data, use technology, manufacture parts, and share ideas. However, for some fields, and specifically for the field of automation, this solution gives rise to a new problem: having a real system to analyze and control.

To solve this issue, one possible option is the use of virtual laboratories, see Goodwin et al. (2010) and references therein. The main problem with virtual laboratories is that they are usually set up so that the user only has to enter the desired test guidelines, as they do not have physical access to tackle the problem from scratch. Therefore, the student will not learn how to manage a real-life system, so, they present the same problem that was discussed for the laboratory practices in which simplified systems were used to be able to focus the practice on the question under study within the available schedule.

Recently, another option has emerged thanks to the exponential reduction in the price of electronic components: all necessary devices are accessible at such a reduced price that each student could acquire his or her own equipment. In other words, each student could have a laboratory in his own home, with physical access and without a limited timetable. For example, a two-wheeled self-balancing robot is presented in Juang and Lum (2013), an example of a ball and beam is shown in Rashied et al. (2016), a magnetic levitator is built to teach control in Lilienkamp and Lundberg (2004), and a self-balancing mobile robot is presented in Gonzalez et al. (2017); Borja et al. (2020). Some of these platforms have become commercial products, such as TCLAB, a low cost temperature control system, Park et al. (2020).

This article presents the speed control of a CPU fan using an Arduino. Using this lab equipment, students can learn basic mechatronics and control concepts at the undergraduate level, including hardware integration, filtering, step

⋆ This work was supported by the Plan Propio of the University of Seville and by the Agencia Estatal de Investigación (AEI)-Spain under Grant PID2019-106212RB-C41funded by MCIN/AEI/ 10.13039/501100011033.

response modeling, frequency response modeling, second-order systems, Nyquist stability, disturbance rejection, PID control, anti wind-up methods and steady-state error.

The sampling time will be guaranteed by temporary interruptions. Interrupts associated with the change of a pin will be used to measure the speed and they will also have to program a filter to have an acceptable measurement of the speed (signal processing). The system can be approximated by a stable first-order system with a delay. This practice is interesting as a first contact with sensors, actuators and control system.

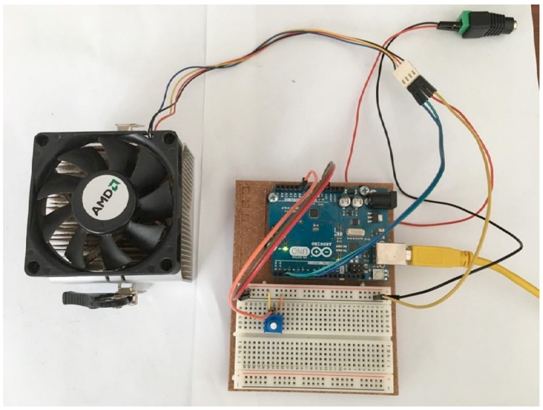
Fig. 1. CPU fan speed control laboratory equipment with potentiometer.

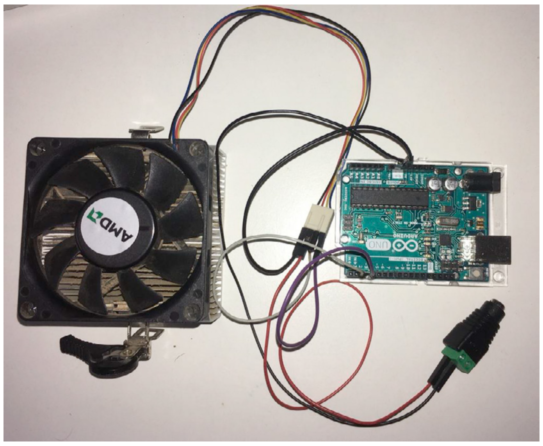
Fig. 2. CPU fan speed control laboratory equipment without potentiometer.

This article will be organized as follows: In Section I the hardware prototype is presented. The control scheme, including how to estimate the fan speed and recover experiment data, is described in Section II. In Section III, different control exercises implemented using the laboratory equipment in a second-year control course are presented. The paper ends with some concluding remarks.

The PWM signal that controls the speed of the fan can have values $\in [0, 255]$ [1] will be displayed in this interval. The rotational speed of the fan ω will be measured in ps ($pulses/second$) as is the way that is going to be measured. The time is measured in seconds.

[1] The frequency of the PWM signal is $490H_z$. If the period is divided in 255 parts, the number of parts that the signal have 5V is the value the signal; the rest are equal to cero.

All the codes used for the generation of the figures can be found in Github.
https://github.com/DocMuerde/SpeedFanControl

2. PROPOSED LABORATORY EQUIPMENT

The speed control of a CPU fan equipment has been designed a way such that it can be built by students without prior electronics knowledge using low cost hardware. Students need:

- A 4-wire CPU fan with an internal encoder and a pulse-width-modulation (PWM) speed control signal.
- An optional 10K potentiometer.
- A 12V power supply.
- Arduino microcontroller with the corresponding USB cable.
- A personal computer to program the Arduino, provide the power to it and show the gathered data.

The CPU fan, potentiometer and the power supply can can be recycled from old equipment, other electronics projects, or they can be obtained for less than 15€. The connections between the different parts of the equipment can be done using a protoboard and do not need soldering.

The key component of the proposed lab equipment is the CPU fan. Most commercial CPU fans are already prepared to be controlled by a digital PWM signal, and although fans run at 12V, it is possible to control them using the 5V PWM signal of the output DC signals of Arduino microcontrollers [2] . In addition, CPU fans have an internal encoder, that typically provides 1 $pulse/revolution$, that we will use to estimate the motor speed, so we already have everything we need to control the speed of the fan. Figures 1 and 2 show the CPU fan used in the results presented in this work.

We will use the potentiometer as a voltage divider. Connecting one pin to 5V, the other to GND and the intermediate pin to one of the analog inputs of the Arduino. In this way we will have a number ranging from 0 to 1023 (since the analog input of the Arduino is 10 bits) controllable by the user. This analog input will be used to set the input signal in manual control mode and the reference in automatic control mode. A potentiometer can be seen in the figure 3.

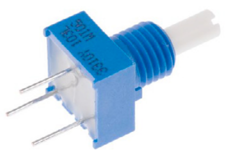
.

Fig. 3. Potentiometer.

This device is optional. The input signal in manual mode and the reference in closed-loop can be set directly by software generating the appropriate signal trajectory for the experiment. Most of the figures presented in the results section were obtained in this way. We include the

[2] The resistance between PWM pin and GND $R \in (5K - 10K)$, so, the intensity provided by the corresponding Arduino digital pin $\in (0.5 - 1)mA$ that is smaller than the maximum intensity that a digital pin can provide (20mA), then, no additional electronics are required

potentiometer because in our experience, students often understand better the notion of external inputs when they can modify them at will.

With respect to the power supply, it is needed to power the CPU Fan. Any 12V DC power supply will do, the only thing to keep in mind is that an adapter is needed to connect the power supply to the rest of the circuit.

We are going to use the Arduino to process the information coming from the encoder and generate the PWM signal needed for the motor to rotate at the reference frequency. In addition, it will output through the serial port, the reference, the current speed, and the control action. These three variables can be displayed using the *Serial Plotter* tool provided with the Arduino IDE.

For this setup we have used an Arduino Uno, but it can be any Arduino. It is a microcontroller based on an ATmega328P. It has 14 digital pins (6 of them PWM) and 6 10bit analog inputs. The speed of the microcontroller is $16MHz$. The Arduino Uno board contains all additional electronics to be able to use the resources of the microcontroller.

The electrical circuit to be assembled including the potentiometer is very simple as it can be seen in figure 4. It is very important to be careful that the 12V terminal of the power supply is not connected to the Arduino in any way, as it could damage it. It is also essential to connect the ground of the power supply to the ground of the Arduino. The Arduino is powered by the computer through the corresponding wire (is not depicted on the figure).

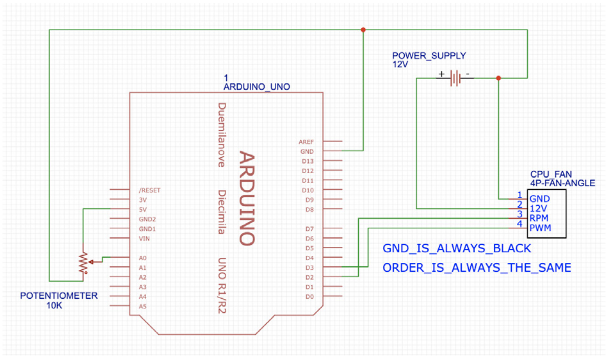

Fig. 4. Connections Arduino-fan. Optionally, a potentiometer can be also used

3. PROPOSED SPEED CONTROL SCHEME

The control scheme is a standard regulation problem in which the control objective is to regulate the CPU fan speed to a given reference by modifying the fan PWM input signal. The CPU fan speed will be estimated from the encoder signal read using one of the Arduino digital inputs. The PWM signal is controlled directly by one of the Arduino PWM digital outputs. The Arduino will implement a standard discrete time control scheme with a given sampling time. Figure 5 shows the two blocks that the student must program in Arduino.

Block 1 is the controller, which is executed once every sampling time. It receives the reference read from the analog input connected to the potentiometer and the fan speed estimated in Block 2. In the exercise examples, a discrete-time PID controller was implemented

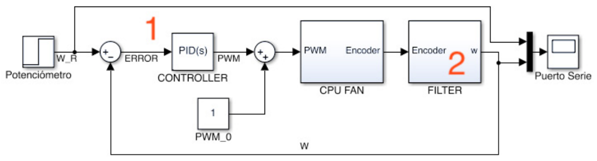

Fig. 5. Control scheme.

Block 2 estimates the motor CPU fan speed ω from the digital input signal connected to the encoder.

To implement this control scheme, the student needs to learn how to deal with interrupt signals in Arduino. Temporary interrupts are used to implement the sampled time control scheme with a sampling time of $100ms$ [3]. In order to measure the fan speed, an interrupt is associated with the change of state of the digital input corresponding to the encoder. Every time the interrupt occurs, a counter is increased. Every sampling time this counter is read. This value can be used to obtain the speed np_{100} in pulses per second as the increment in the counter times 10, that is,

$$np_{100}(k) = 10(\text{counter}(k) - \text{counter}(k-1)) \quad \text{pulses/s}$$

However, this signal has a very low precision: Setting the maximum speed, that is, with the PWM output signal of the Arduino set to 255, the CPU fan only generates 15 pulses, and with the minimum speed, that is, with the PWM output signal of the Arduino set to 0, the CPU fan only generates 5 pulses [4]. With this speed measurement precision, it is impossible to implement a speed controller.

In addition, the CPU fan presents first-order stable dynamics. To artificially add high-order dynamics and to obtain a more smooth measured signal, we propose to filter the speed measurement, that is, instead of controlling the estimated speed from the latest measurements, control the mean speed of the last N sampling times.

The idea is create a buffer that stores the last N measurements of np_{100}. In each sampling period, a new measurement will be added and the oldest one will be discarded. In each sampling period, the values of the N measurements in the buffer will be summed and that will be the measure of the mean CPU fan ω over the filter window N considered in pulses per second.

$$\omega(k) = \frac{10}{N} \sum_{i=0}^{i=N-1} np_{100}(k-i) \qquad (1)$$

To carry out experiments, the students need to be able to store the signal trajectories of the speed, reference and input signals. There are different solutions to this problem. One option is to use the Serial Plotter tool in Arduino IDE. Serial plotter receives data from the Arduino from the USB cable and is able to visualize the data received

[3] The sample time have to be chosen taking this two issues into account. If the sample time is small the range of pulses each sample time is going to be decreased lowering the quality in the measurement while if it is chosen bigger the dynamic is not sample properly

[4] These values depend on the CPU fan used, but in general, the signal will have a low precision for any commercial CPU.

both in a text display or in a graph. The serial plotter is able to receive multiple signals simultaneously.
In Figure 6 the response of the system when the value of the PWM signal changes from 75 to 175 is show for $N = 10$. A new measurement of w is plotted each sample time.

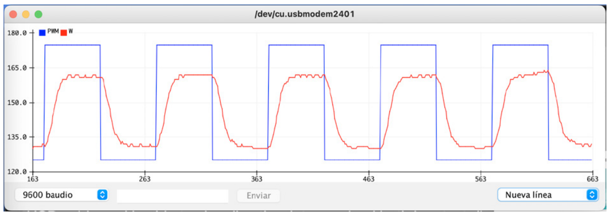

Fig. 6. Step response with $N = 10$ using *Serial Plotter* of Arduino.

As it can be seen in figure 6, it is very difficult to measure the parameters of the response just using the *Serial Plotter* of Arduino. However, the data received is not only visualized in graph form, but it is also shown in a text display, the Serial Monitor. In the exercises carried out, the students export this data to Matlab, copying the text direcly from the *Serial Monitor* of Arduino into a matlab script file which creates the corresponding signal vectors. With Matlab, it is possible to plot, measure, and use the experimental results in a more efficient way. In Figure 7 same response of the system when the value of the PWM signal changes from 125 to 175 is show for $N = 10$ in a figure generated with Matlab.

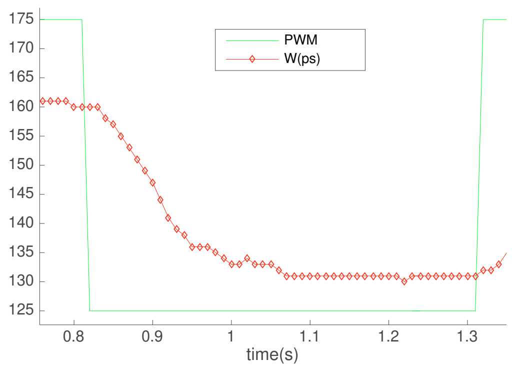

Fig. 7. Step response with $N = 10$ using Matlab to plot the data received using *Serial Monitor*.

4. CONTROL EXERCISES EXAMPLES

The presented lab equipment can be used to implement many control exercises including step response identification, frequency response modeling, second-order systems, Nyquist stability, disturbance rejection, PID control, anti wind-up methods and steady-state error. It was first developed to teach students doing their final degree and master projects how to implement Arduino-based controllers. It is a simple application that lays very well the foundations for developing more complex control systems.

In the 2020-21 course, the CPU speed control equipment was used as training for students of the second-year course "Automatic Control" of the University of Seville's telecommunications engineering degree who did a voluntary project which consisted in building and controlling a line-following robot Alvarado et al. (2021). The students' feedback was very positive, and in the course 2021-22 this project has become mandatory and the control laboratory sessions will use the CPU fan speed control system. We present next some results of different exercises that the students will address in that course.

4.1 Steady state response

The steady state response of the CPU fan can be obtained by applying different constant PWM inputs to the motor. Figure 8 shows the speed in steady state for different values of the input signal. Note that the PWM signal takes values from 0 to 255, while the speed takes values from 70 to 210 pulses per second.

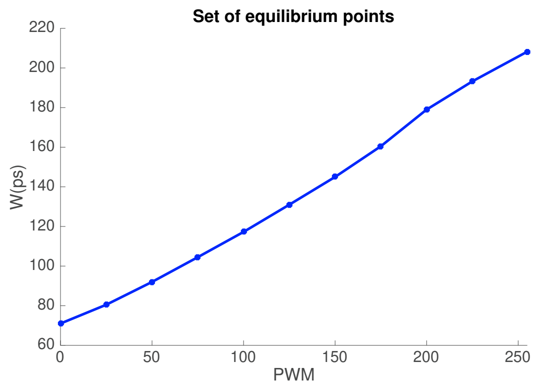

Fig. 8. Set of equilibrium points.

4.2 Step response identification

The system can be approximated as a first-order system with a delay or with a slow pole. In figure 9 the response of the system when the value of the PWM signal changes from 125 to 175 is show for $N = 20$. The values of the PWM signal are controlled by program loaded in the Arduino. The step response shows that the system presents a small delay artificially created by the algorithm that computes the filtered revolutions per second. This delay can be increased just increasing the filter window N, used for estimating the FAN speed. As N increases, the effect of the filter adds more inertia to the system, which can be modeled with a delay or fast poles.

4.3 Frequency response

The frequency response of the system can be analyzed generating sine PWM signals around the operation point PWM_0, ω_0 [5]. Figure 10 shows the response of the system to a sine input of $1 rad/s$. This figure is obtained from the data gathered from *Serial Monitor* as was explained in section 3. Students can learn how to obtain experimentally the frequency response of the system, carrying out different experiments with different frequency inputs. Figure 11 shows the bode diagram obtained from seven different experiments.

[5] $PWM_0 = 125$, $\omega_0 \simeq 132.1$

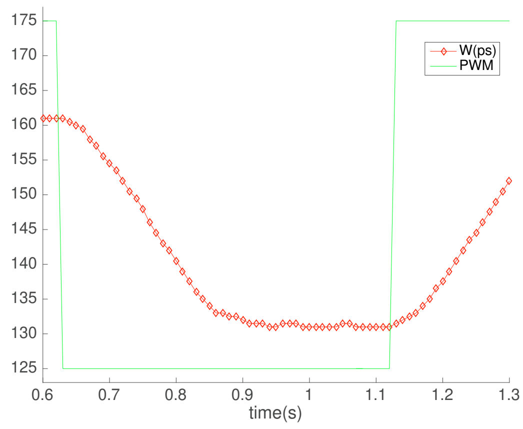

Fig. 9. Step response with $N = 20$.

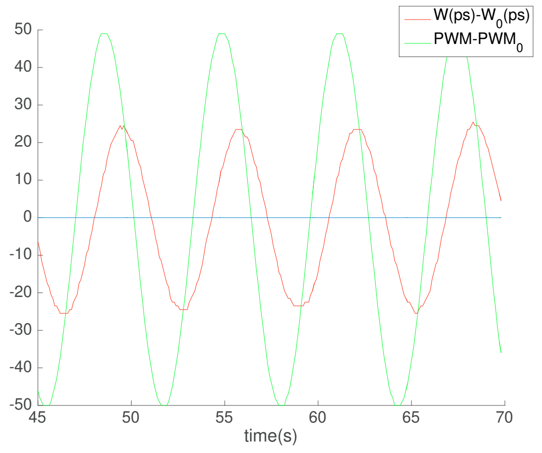

Fig. 10. Frequency response for a 1 rad/s sine input.

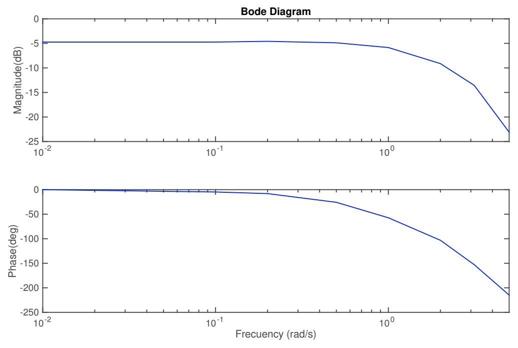

Fig. 11. Experimental Bode diagram obtained from 7 experiments.

4.4 PID speed control

Using the proposed control scheme, standard PID discrete-time control laws can be implemented. The 100ms sampling time is fast enough to allow for a continuous time analysis.

Figures 12 show a capture of the Arduino Serial Plotter tool showing the evolution of the reference, the motor speed, and the PWM signal that controls the motor. For this example, the reference was set by the potentiometer. In this test, various changes in the reference are shown.

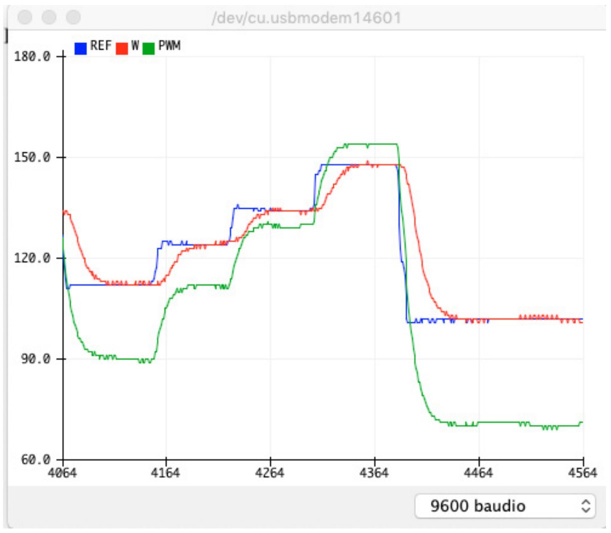

Fig. 12. Closed-loop experiment.

4.5 Disturbance rejection

The laboratory equipment can be modified to include disturbances. Figure 13 shows the response of the controller when the system is subject to a disturbance. In this experiment, a sheet of paper was placed on top of the fan obstructing the passage of air. It can be seen that the control action varies to compensate for the effect of the disturbance. In the last part of the test, the sheet of paper is removed again.

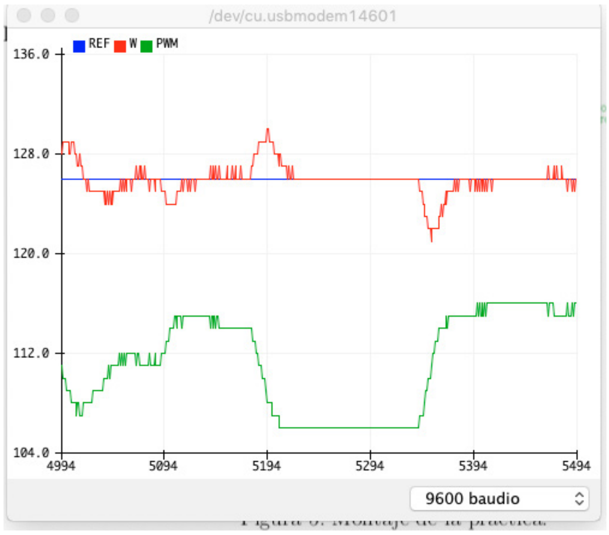

Fig. 13. Response of closed-loop system when the CPU fan is subject to a disturbance using a sheet of paper between samples 5180 and 5350.

4.6 Wind-up

The PWM input of the CPU fan is constrained by the limits of the Arduino PWM output frequency. In particular, the output PWM signal can take values from 0 to 255.

This implies that wind-up can occur for speed references larger than the maximum speed.

Figure 14 shows the fan controlled by a PI. The wind-up effect is visible after the third change of reference, which is higher than the maximum speed.

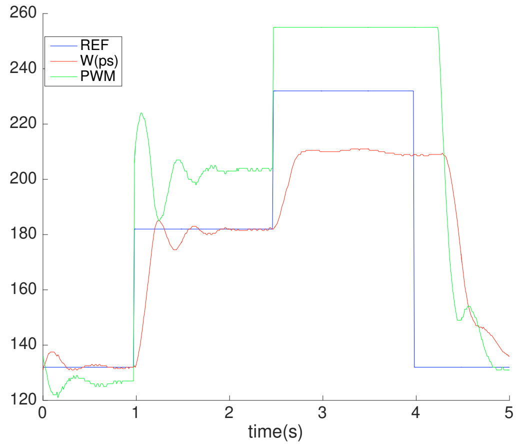

Fig. 14. Wind-up effect in a PI controller.

A saturation in the integral term of the PID can be tune in order to counteract this effect.

4.7 Stability of feedback systems and Zieger Nichols

The CPU fan can be modelled as a first-order system with a small delay. This implies that classical process control PID tuning methods such as Zieger-Nichols can be used and that the stability issues that may appear in feedback systems can be demonstrated.

To implement the second Zieger-Nichols method, it is necessary to find the critical gain of a proportional controller that provides an unstable closed-loop system with complex poles. This gain can be found by trial and error by finding the gain that provides oscillatory responses to step references. Figure 15 shows the system controlled by a proportional control in the stability limit. As it can be seen, the input is not saturated, which makes this system appropriate for this exercise. From this experiment, Zieger Nichols rules can be applied.

5. CONCLUSIONS

This article presents a low cost lab equipment, the speed control of a CPU fan using an Arduino. Using this lab equipment, students can learn basic mechatronics and control concepts at the undergraduate level. Although the laboratory equipment presents a simple control problem (filtered first-order system), it has as a great advantage with respect to other laboratory equipment, which is that due to its cost and simplicity of assembly, it is possible for each student to acquire their own equipment to work at home and solve the different problems that proposed. This makes it possible to carry out practices with greater depth, including a more complete analysis than would be possible in the laboratory in a limited time session. In addition, the student work becomes multidisciplinary, as they need to apply not control theory, but also electronics, programming and signal processing.

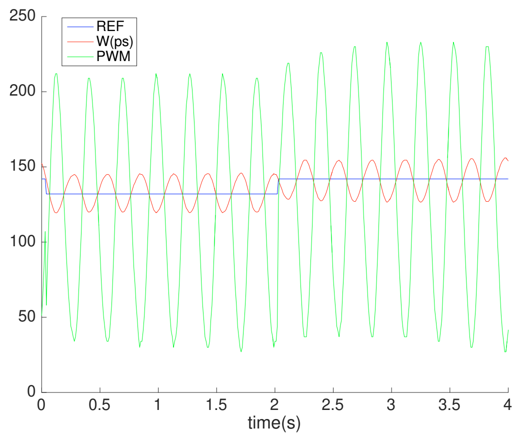

Fig. 15. Identification of the critical gain and period.

REFERENCES

Alvarado, I., Borja, J.A., Salas, F., and Muñoz de la Peña, D. (2021). Aprende fundamentos de control construyendo un siguelíneas. 248–253.

Borja, J., Alvarado, I., and Muñoz de la Peña, D. (2020). Low cost two-wheels self-balancing robot for control education powered by stepper motors. *IFAC-PapersOnLine*, 53(2), 17518–17523.

Gonzalez, C., Alvarado, I., and Muñoz de la Peña, D. (2017). Low cost two-wheels self-balancing robot for control education. *IFAC-PapersOnLine*, 50(1), 9174–9179.

Goodwin, G.C., Medioli, A.M., Sher, W., Vlacic, L.B., and Welsh, J.S. (2010). Emulation-based virtual laboratories: A low-cost alternative to physical experiments in control engineering education. *IEEE Transactions on Education*, 54(1), 48–55.

Juang, H.S. and Lum, K.Y. (2013). Design and control of a two-wheel self-balancing robot using the arduino microcontroller board. 634–639.

Lehmann, M., Christensen, P., Du, X., and Thrane, M. (2008). Problem-oriented and project-based learning (popbl) as an innovative learning strategy for sustainable development in engineering education. *European journal of engineering education*, 33(3), 283–295.

Lilienkamp, K. and Lundberg, K. (2004). Low-cost magnetic levitation project kits for teaching feedback system design. *Proceedings of the American Control Conference.*

Park, J., Martin, R.A., Kelly, J.D., and Hedengren, J.D. (2020). Benchmark temperature microcontroller for process dynamics and control. *Computers & Chemical Engineering*, 135, 106736.

Perrenet, J.C., B.P. and Smits, J. (2000). The suitability of problem-based learning for engineering education: theory and practice. *Teaching in higher education.*, 5(3), 345–358.

Rashied, Z., Hamees, M., Hassan, M.U., Hameed, S., and Khatri, N. (2016). Real time implementation of robust pid controller for stabilization of ball balancing beam. *International Journal of Conceptions on Information Technology and Computing*, 4(2), 6–9.