Design and implementation of a low-cost remote tank control system for engineering and technology education

Christian Nomesqui Galvis
Dept. of Technology in Electronics
Corporación Universitaria Minuto de Dios
(UNIMINUTO), Bogotá, Colombia
cnomesquiga@uniminuto.edu.co

Harold Chavarro Medina
Dept. of Technology in Electronics
Corporación Universitaria Minuto de Dios
(UNIMINUTO), Bogotá, Colombia
hchavarrome@uniminuto.edu.co

Jonathan Álvarez Ariza
Dept. of Technology in Electronics
Corporación Universitaria Minuto de Dios
(UNIMINUTO), Bogotá, Colombia
jalvarez@uniminuto.edu

***Abstract*—In this paper, we describe the design and implementation of a low-cost tank control system for engineering and technology education which arose according to the challenges for education produced by the pandemic of COVID-19. The interest with this investigation was to develop in a short time a remote laboratory for control systems education wherein the students were able to design and implement their digital classic controllers (P, PI, and PID). In this way, students can put into practice the different learned concepts in control systems with hands-on activities regardless of the limitations in laboratory practices generated by the pandemic. In the remote laboratory, the students can program their digital controllers using the programming language Python and performing the tasks that are commonly employed in control systems such as plant identification, modeling, controller design, implementation, and debugging. From the technical perspective, the laboratory is composed of two tanks whose level is measured by an analog ultrasonic sensor and it is controlled by a Raspberry Pi with a set of motor pumps. The laboratory is embedded entirely in a web engine which allows the students to access from their homes or in any location. At last, we expose the different experiments with the plant and the control system to illustrate the features and potentialities of the proposed remote laboratory.**

***Index Terms*—Remote laboratory, control systems, engineering education, technology education, python language.**

I. Introduction

THE pandemic of COVID-19 has originated a set of educational challenges in higher education and it has transformed the in-person spaces of learning and classrooms towards virtual or remote settings. Due to the restrictions and lockdowns decreed by local policies to preserve the health of students, educators, and administrative staff in the major part of the countries, the experimentation, laboratories, hands-on activities, and problem-solving developed by the students have been seriously affected. Laboratories represent the basis of the experimentation in engineering and they have formed an active part of the curricula. Students experiment and construct their meanings with hands-on activities, which are the essence of science learning [1]. Aside from these factors, the concept of Emergency Remote Teaching (ERT) has taken on an important relevance in education. ERT differs from online learning since it is a temporal method of learning and teaching due to crisis circumstances [2], [3]. As Hodges et al. [2] describe, the primary objective of the ERT is not to recreate a robust educational system but to be a method to support the learning needs of the students under the crisis period. In addition, according to UNESCO [4], the socioeconomic factors, anxiety, problems with Information and Communication Technologies (ICT) and internet connectivity, social isolation, and keeping the regular scheduling are impacts produced by the pandemic in the world with special emphasis in Iberoamerica and North America regions.

So, in order to address in part these problems, it is needed the development of methodologies and technologies that help to support the educational process of the students. In this context, this work describes the design and development of a low-cost remote tank control system for engineering and technology education. The laboratory is composed of a tank (plant) whose level is sensed by an analog ultrasonic sensor and controlled by a set of motor pumps that are interfaced to a Single-Board Computer (Raspberry Pi V4). The experiment counts with a real-time video provided by a Raspberry Pi camera and at any moment the students can change the parameters of the designed controller in the Python programming language. We have chosen this language because of its versatility, scalability, and ease of learning. Some studies in Computer Science (CS) and in the field of control systems have demonstrated that this language fosters learning and engagement of the students in the courses [5], [6], [7].

According to the aforementioned, this work is divided as follows: Section II describes the platform to technical level in the components of hardware and software. Section III shows

an experiment to control the level of the proposed control plant with the respective results. Finally, section IV and V outlines the discussion, conclusions, and further work of this proposal.

II. Laboratory Description

A. Hardware setup

Table I depics the main hardware components of the laboratory which in their majority are of low-cost. However, these components are robust to accomplish with the requeriments of the experiment in control systems education.

TABLE I
HARDWARE SETUP FOR THE REMOTE LABORATORY.

Component	Quantity/Description	Cost (USD)
Control plant	(2) Acrylic tanks: height=50cm, area=301.7cm^2	37.5
Raspberry Pi 4 (2GB RAM)	(1) Single-Board Computer Raspberry Pi V4	40
Camera	Raspberry Pi camera 8MPx	30
Motor pump	(2) 12V, motor pumps	30
US-016	(1) Analog ultrasonic sensor (Range 4cm-300cm)	4
MCP6004	(1) Rail-to-rail operational amplifier for signal conditioning	2
Switching power supply	(1) Power supply for the motor pumps, sensor, and signal conditioning	15
L298	(1) Motor Driver	2
ADS1115	(1) Analog to Digital Converter (ADC)	5
Total (USD)	**-**	**165.5**

Then, with these components was designed and implemented the control system. The MCP6004 was used for signal conditioning of the sensor US-016 concerning gain and offset, getting a linear gain $H = 0.048\frac{V}{cm}$. For instance, if the level of the tank is 30cm, then the output voltage of the sensor will be $V_o = 0.048\frac{V}{cm} \cdot 30cm = 1.44V$. Fig.(4) shows the final design considering the previous aspects.

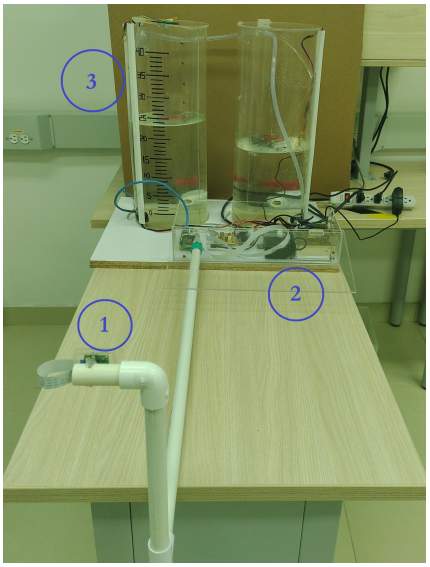

Fig. 1. Appareance for the final plant design. 1. Camera of the experiment. 2. Raspberry Pi, signal conditioning, and power interface for the motor pumps. 3. Control plant (tank system).

B. Software setup

For the software setup, we developed a web engine console for the students directly write the code in Python. In the interface, it was utilized the code editor ACE available at (https://ace.c9.io/) to highlight the different statements in the programming language to the student recognize them. Concerning the laboratory access, it was employed the Python packages *Flask* and *Flask logging* [8], [9] to manage the user login and the actions to take when a student sends a code for the experiment. The student or user must login with an accredited email and password.

One of the critical elements in the laboratory construction was video streaming. To test the latency of the video streaming for the users or students, we used some servers and gateways such as Nginx+HLS protocol, Node-Media server+HTTP FLV, and Janus Media server [10], selecting the last one due to the low latency experimented which was less of 1 sec. The video is encoded by the FFmpeg tool [11] in the Raspberry Pi and sent through Real-Time Streaming Protocol (RTSP) or User Datagram Protocol (UDP) for further processing in a server that establishes the communication between the user and the laboratory. Besides, a Virtual Private Server (VPS) manages the connection between the clients (students) and the experiment and also it processes the video through the Janus media server and the standard WebRTC. Similarly, the different codes that the students develop for their controllers are transferred using Websockets in Python with the package *Python-socketio* [12].

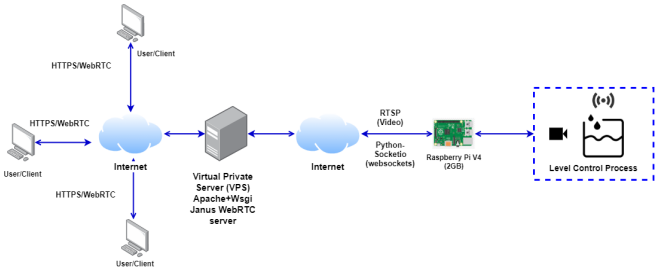

Fig. 2. Overall scheme for the network architecture of the remote laboratory.

III. Design and Experiments with the Plant and Control System

In this section, it is shown an experiment to control the level of the proposed control plant (tanks). Also, we followed the steps that typically one student performs concerning the controller design and the usage of the platform.

A. Identification

In this stage, the student identifies the plant applying a step source that can change over time. In addition, our starting point was the differential equation for a basic level plant which is defined as follows:

$$A\frac{dh(t)}{dt} = Q \tag{1}$$

Where A is the tank area, $h(t)$ de change in the level, and Q is the flow of the liquid, in this case in $\frac{cm^3}{s}$. After the plant identification experiments with a reference $(h(t) = 40cm)$ and the transformation of the equation (1) to Laplace domain (s), the function transfer for the plant was $G(s) = \frac{47.7}{301.7s}$. In addition, for the previous conditions, we identified the liquid flow provided for the motor pumps as $Q = 47.7\frac{cm^3}{s}$ with a voltage $V = 12V$. Besides, the identified function transfer for the sensor (US-016) with the signal conditioning was $H(s) = 0.048$. With these values, we proceeded to design and implement the controllers (P, PI, and PID). Fig.(3) illustrates the step response for the plant in MATLAB with $Q = 47.7\frac{cm^3}{s}$.

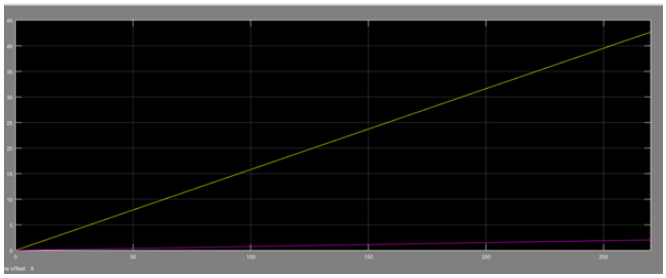

Fig. 3. Step response for the control plant (yellow-unstable system).

B. Controller design

For this part, we employed the MATLAB tool (*SISOTOOL*) that is an utility to design and develop controllers or compensators through root locus method. We added the poles or zeros of each controller according to its expression in Laplace transform. For instance, the proportional controller has the expression $C(s) = K_pE(s)$, where $E(s)$ is the error of the control system. By the same token, the PI and PID controllers have the following expressions in Laplace domain: $C_{PI}(s) = K_pE(s) + \frac{K_I}{s} = K_p\frac{(s+a)}{s}, a = \frac{Ki}{Kp}$ and $C_{PID}(s) = K_pE(s) + \frac{K_I}{s} + K_ds = K_p\frac{(s+a)(s+b)}{s}$. In these equations K_p, K_I, K_D are the proportional, integral, and derivative constants, respectively. The design requeriments for the control system in the experiments with a setpoint $R(s) = 40cm$ were a settling time ($t_s < 255sec$) and a Percentage Overshoot (PO) $\leq$10%. For the controllers, we took a feedback structure as it is depicted in Fig.(4).

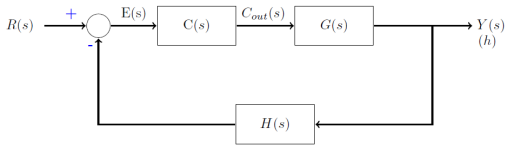

Fig. 4. Control block diagram for the experiment. Control Plant $G(s)$, controller $C(s)$, sensor and conditioning $H(s)$, reference $R(s)$, control output process $Y(s)$.

Because the controllers must be implemented in a Raspberry Pi, it is needed to transform the expressions of the controllers provided by the tool *SISOTOOL* in the Laplace domain towards digital controllers in Z transform. The method can be taught to the students in MATLAB employing the command *c2d* or directly making the left-sided Z transform. The PI controller was converted in Z transform, using both the methods Zero-Order Holder (ZOH) and Tustin (Bilinear transform). Regarding the PID controller was employed the method Tustin. This process yielded to the following controller's expressions in Z transform and in *difference equations* ready for Python implementation:

- $C_p(z) = 100E(z) \rightarrow C_p(kT) = 100e(kT)$
- $C_{PI}(z) = \frac{120z-119.4}{z-1}E(z) \rightarrow C_{PI}(kT) = 120e(kT) - 119.4e(kT-1) + C(kT-1)$
- $C_{PID}(z) = \frac{7.754z^2-15.47z+7.716}{z^2-1}E(z) \rightarrow C_{PID}(kT) = 7.754e(kT)-15.47e(kT-1)+7.716e(kT-2)+C_{PID}(kT-2)$

Where *C(z) or C(kT)* is the output of the controller, *E(z)* or *e(kT)* is the discretized error of the control process. *(kT-1)* and *(kT-2)* represent a delay of one or two samples depending on the variables of *error* and *controller's output*. For each controller, we have chosen a sampling time of $T = 0.6sec$. With the latter difference equations, we deployed and debugged the controllers in Python language.

C. Controller implementation

To implement each controller, the mentioned difference equations were converted to Python language. For that, firstly, we employed the package Adafruit ADS11X5 for the ADC which operates through I^2C protocol. This package or library provides the sensed voltage by the ultrasonic sensor (US-016) with its respective signal conditioning. Secondly, with this voltage, we calculated the error $e(kT)$ and we deployed the controllers P, PI, and PID. Thirdly, the output controller was normalized to meet the requirements for the Pulse Width Modulation (PWM). Finally, we evaluated the performance and feasibility of each controller in the remote experiment as it is depicted in Fig.(5). Moreover, each one of the codes implemented for the controllers can be found at [13]. In some cases, the PI controllers were limited in their outputs (anti-windup) to improve the plant response and the settling time (t_s).

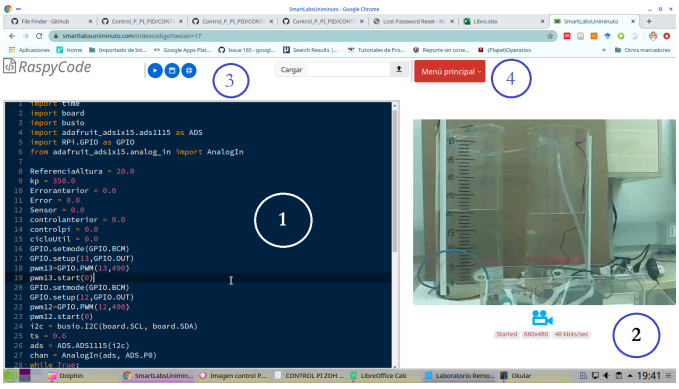

Fig. 5. Example of controller implementation for the remote experiment. 1. Working area for Python programming. 2. Real-time video for the experiment. 3. Buttons for play the experiment, save the code and help. 4. Main Menu to plot variables, see console output, and close session.

To evaluate the better controller for the plant that is a common task developed by the students, we measured the

settling time (t_s) for each controller with different values of setpoints or references. This information is condensed in Fig.(6). For the plant, in this case, the PI controller through ZOH method provides better efficiency for the control system purposes.

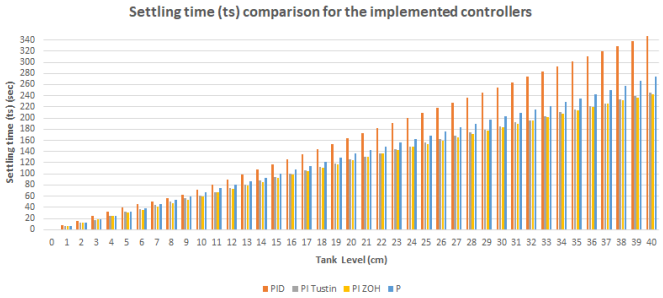

Fig. 6. Performance comparison in terms of settling time (t_s) for each controller implemented.

D. Controller debugging

Once the previous controllers were designed, we modified the values of the proportional constant (K_p) and integral constant K_i to get a better response for the control system. For instance, Fig.(7) depicts several values of K_p for the PI through ZOH method. Similarly, we fixed some errors that were evidenced in Python language regarding the implementation of the controllers.

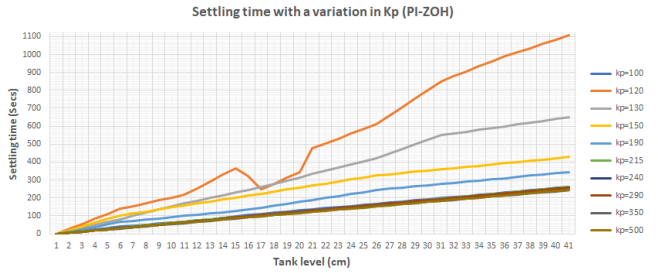

Fig. 7. Comparison of the settling time (t_s) with a variation of the (K_p) value for the PI controller through ZOH method.

IV. Discussion

Our interest in this experiment was to show how a low-cost control system can be designed and implemented for educational purposes in higher education. As mentioned, the COVID-19 pandemic has represented new challenges in education and one of them is to provide high-quality education and continuity in it despite the lockdowns and mobility restrictions. Hence, the experiment proposed in this work started from a curricular reflection in order to offer different possibilities to the students for experimentation through hands-on activities from their homes. Also, we followed the typical steps that one student can perform with the control system in order to test the robustness and efficiency of the experiment. The server described in Fig.(2) could be replaced, for instance, by another Raspberry Pi, reducing the costs of the implementation of the experiment. The critical requirements are an internet connection and a camera for real-time video. Furthermore, in the experiment, we employed Python language due to its versatility and ease to learn which is an advantage for the students who want to implement their controllers and learn about analog and digital control systems. Although in this experiment we implemented classic controllers (P, PI, PID), other techniques such as state-space, fuzzy logic, frequency domain, among others could be developed with this control plant.

V. Conclusions

In this study, we described the design and development of a low-cost remote laboratory for control systems education. In the remote laboratory, we employed common elements such as Raspberry Pi, acrylic tanks, ultrasonic sensors, and so on. Our aim with his work was to illustrate that it is possible to create low-cost control plants that support the learning process of the students despite the lockdown periods and the restrictions in hands-on practices produced by the COVID-19 pandemic. Besides, in the development of the laboratory, we used open-source software, e.g., in the Janus media server for the real-time video or in the multiple python packages to deploy the Apache server which provides access to the students in the experiment. Further work will be focused on the development of new low-cost plants for the students to experiment with. Also, we will construct an educational survey to evaluate the impact of these experiments on the learning process of the students.

References

[1] N. J. Nersessian, "Conceptual change in science and in science education," *Synthese*, vol. 80, no. 1, pp. 163–183, 1989.

[2] C. Hodges, S. Moore, B. Lockee, T. Trust, A. Bond *et al.*, "The difference between emergency remote teaching and online learning," *Educause review*, vol. 27, no. 1, pp. 1–9, 2020.

[3] M. D. Rahiem, "The emergency remote learning experience of university students in indonesia amidst the covid-19 crisis," *International Journal of Learning, Teaching and Educational Research*, vol. 19, no. 6, pp. 1–26, 2020.

[4] UNESCO, "Covid-19 and higher education: Today and tomorrow," 2020.

[5] A. Vergnaud, J.-B. Fasquel, and L. Autrique, "Python based internet tools in control education," *IFAC-PapersOnLine*, vol. 48, no. 29, pp. 43–48, 2015.

[6] A. Jayal, S. Lauria, A. Tucker, and S. Swift, "Python for teaching introductory programming: A quantitative evaluation," *Innovation in Teaching and Learning in Information and Computer Sciences*, vol. 10, no. 1, pp. 86–90, 2011.

[7] Á. Hoyo, J. L. Guzmán, J. C. Moreno, and M. Berenguel, "Teaching control engineering concepts using open source tools on a raspberry pi board," *IFAC-PapersOnLine*, vol. 48, no. 29, pp. 99–104, 2015.

[8] (2021) Flask python package. [Online]. Available: https://flask.palletsprojects.com/en/2.0.x/

[9] (2021) Python-socketio package. [Online]. Available: https://python-socketio.readthedocs.io/en/latest/

[10] A. Amirante, T. Castaldi, L. Miniero, and S. P. Romano, "Janus: a general purpose webrtc gateway," in *Proceedings of the Conference on Principles, Systems and Applications of IP Telecommunications*, 2014, pp. 1–8.

[11] F. Korbel, *FFmpeg Basics: Multimedia handling with a fast audio and video encoder*. Frantisek Korbel, 2012.

[12] (2021) Flask loggin python package. [Online]. Available: https://flask.palletsprojects.com/en/2.0.x/logging/

[13] (2021) Repository for the implemented controllers (p, pi, pid). [Online]. Available: https://github.com/ChristianNomesqui/Control_P_PI_PID