International Journal of Combinatorial Optimization Problems and Informatics, 17(1), Jan-April 2026, 20-33. ISSN: 2007-1558.
https://doi.org/10.61467/2007.1558.2026.v17i1.1200
www.editada.org

Electronics and Programming for Bioprocess Control in Biotechnology Engineering: Accessible Solutions for Industry 4.0

Marco Antonio Márquez-Vera[1], Ricardo Calderón-Suárez[2], Evelin Gutiérrez-Moreno[1], Ocotlán Díaz-Parra[1], Carlos Antonio Márquez-Vera[3]
[1] Mechatronics, Polytechnic University of Pachuca
[2] Computer Science, Autonomous University of Hidalgo State
[3] Chemical Engineering, Universidad Veracruzana
Emails: marquez@upp.edu.mx, ricardo_calderon@uaeh.edu.mx, evgutierrez@upp.edu.mx, ocotlan_diaz@upp.edu.mx, carmarquez@uv.mx

Abstract. This article examines the importance of teaching electronics and process control in biotechnology engineering program, particularly in the context of Industry 4.0. With the rise of automation and Internet of Things (IoT), it is essential to equip students with practical skills. However, economic constraints and the limited access to specialized software in developing countries pose challenges. This study demonstrates how affordable microcontrollers like the ESP32 and open-source software offer an effective solution, enabling hands-on learning without compromising the quality of education. Specifically, the ESP32 has Wi-Fi connectivity, enabling online control and monitoring, as well as data storage in a database. Practical projects such as data acquisition, signal conditioning, and actuator control provide real-world experience, while project-based evaluation and detailed rubrics enhance students' understanding and performance, preparing them for the modern workforce.
Keywords: Electronics; process control; engineering education; Internet of Things.

Article Info
Received August 13, 2025
Accepted August 22, 2025

1 Introduction

The use of computers and automation is widespread in Industry 4.0. (Saravanan et al., 2022). Industrialized countries have the necessary infrastructure to integrate these concepts and technologies into universities, preparing students for their future careers (Faritha et al., 2020). There are some applications to help development countries to have more tools to teach modern concepts and methods in their classrooms, Botero et al. (2016) designed a virtual laboratory to know and manage industrial equipment for chemical engineering students.

Due to economic and management challenges, developing countries face limitations and disadvantages in education (Jhurree, 2005). The use of open-source software offers a solution for teaching various engineering courses (Barry, 2009; Thapa and Gautam, 2021). Numerous applications in engineering utilize open-source software, such as Octave (Sanbatoro et al., 2023; Bertsatos and Chovalopoulou, 2019), Scilab (Cugnet et al., 2023; Vishwanatha et al., 2023), and Python (Caro et al., 2024; Seifrid et al., 2024; Ozdilek et al., 2024), primarily in fields like thermodynamics, robotics, mechatronics, and artificial intelligence.

For biotechnological engineering, private software is available that can be implemented in teaching certain courses. For example, Paoli et al. (2024) used Julia® to develop a course in thermodynamics, highlighting the need for programming skills among students. Another widely-used private software is Matlab®, Zárate-Navarro et al. (2024) employed this tool to teach transport phenomena, successfully covering stages like simulation, and experimental validation. Similarly, Hrnčiřík and Kohout (2024) made a simulator to control fed-batch yeast fermentations using Matlab®.

An interesting tool in Matlab® is SimBiology, which is specifically designed to simulate, model, and analyse dynamic systems, particularly in the pharmaceutical industry. Islam et al. (2024) used this toolbox to model the degradation of dioxins and dibenzofurans. Unfortunately, developing countries often lack the financial resources to purchase licenses for private software and therefore turn to open-source alternatives for teaching.

Python is in particular widely adopted for developing chemical engineering courses. For instance, Domínguez et al. (2021) utilized Jupyter Notebooks to create educational materials, providing examples, notes, and exercises on various topics. Similarly, Caccavale et al. (2023) employed Python to enhance digital skills in chemical engineering students. With this tool students can harness artificial intelligence (AI) to optimize experiments. Kakkar et al. (2021) demonstrated how to build and train artificial neural networks (ANNs) to model adsorption equilibrium datasets.

The World Chemical Engineering Council has recommended incorporating digitalization into chemical engineering education (Feise and Schaer, 2021). Khan et al. (2021) proposed a digital process safety course to be integrated into chemical engineering curricula. They noted that, in control theory, faults can be treated as external disturbances that prevent the proper regulation of process variables.

Laboratory equipment is also crucial for chemical education (Gunasekera et al., 2020). Quive et al. (2021) suggested utilizing readily available materials, such as chalk, pen tubes, and birthday balloons, to conduct experiments for teaching chemical engineering in these regions, given the insufficient resources for science, technology, engineering, and mathematics (STEM) education.

At the Polytechnic University of Pachuca (PUP), a course called Bioprocess Control is offered in the eighth trimester of the Biotechnological Engineering program. The topics covered include Laplace transforms, closed-loop control, PID control, state-feedback, and state observer design. At PUP, the software of choice is Octave, particularly its symbolic and control packages. However, sensors, actuators, and controllers are not available, limiting the opportunity for hands-on experimentation with control theory concepts beyond software simulation.

To develop bioprocess control projects in the classroom, it is essential first to grasp the fundamental principles of control theory and to simulate dynamic systems using Octave. They are defined the actuators as elements that provide energy or apply control signals to the process; they can include heaters, mixers, pumps, aerators, and more. In a practical setting, these elements can be represented by inexpensive and readily available items such as fish tank water pumps, fans, and electric resistors.

The most common and readily available sensors include those for humidity, pH, oxygen, alcohol, carbon dioxide, butane, and temperature. The devices that read measurements from these sensors typically use analog-to-digital converters (ADCs) that give the readings as integer values rather than floating-point numbers.

To motivate students and incorporate IoT, it is recommended to use the ESP32 microcontroller for implementing the computed controller or observer, and data acquisition for process control (Márquez-Vera et al., 2023). The ESP32 includes Bluetooth and Wi-Fi modules, enabling real-time process monitoring and allowing students to observe measurements online and remotely control the bioprocess. This approach also lets students use their smartphones to control or at least automate aspects of the bioprocess. The curricula of bioprocess control given in the eighth four-month period at the PUP is shown in Fig. 1, where the topics are divided into three modules; the first one is an introduction to control theory, principally using transfer functions, and then the second module introduces concepts about state-space representation. Finally, the last module introduce concepts of practical control, especially electronics which is the issue shown in this paper.

2 Digital Conditioning for Sensors

Data acquisition cards and microcontrollers operate within specific voltage ranges. Typically, they accept signals between [0V, 5V]; in some cases, the voltage range is limited to [0V, 3.3V], as with the ESP32 and Arduino Due (Santos and Reis, 2021). Some sensors have output signals that can include negative values. In such cases, the signal can be adjusted using operational amplifiers (Reverter and Areny, 2009). These amplifiers can increase or decrease the signal values, or even add a voltage offset to prevent negative values. This is important because operating with voltage levels higher than specified for the electronic device can damage the circuit. Additionally, if the voltage exceeds 5V or falls below 0V, the electronic card will saturate, leading to incorrect data being recorded (Ahmad et al., 2023).

For example, if we use the pH meter SKU: SEN0161, which operates within a voltage range of [-414.12mV, 414.12mV] for pH values between [0, 14], and it is connected to the ESP32 microcontroller, we can design an amplifier with a gain of

$$A = \frac{3.3V}{2(0.41412V)}, \quad (1)$$

and then add an offset of 3.3V/2 = 1.65V. This ensures that the signal sent to the microcontroller falls within the range [0V, 3.3V], allowing full utilization of the analog-to-digital converter (ADC) scale. It's important to note that a pH of 7 will result in a signal of 1.65V after amplification. Operational amplifiers can also be used as low-pass filters, which help reduce noise in the signal. Without delving into filter design theory, it is generally assumed that frequencies higher than 10 times the signal frequency to be acquired are considered electrical noise. For the SKU: SEN0161, which takes five seconds to output a stable voltage corresponding to the pH value, we can design a second-order filter with a cut-off frequency of 2Hz. The analog filter uses two capacitors and two resistors, and if both resistors and capacitors are identical, the cut-off frequency is

$$f = \frac{1}{2\pi RC}, \quad (2)$$

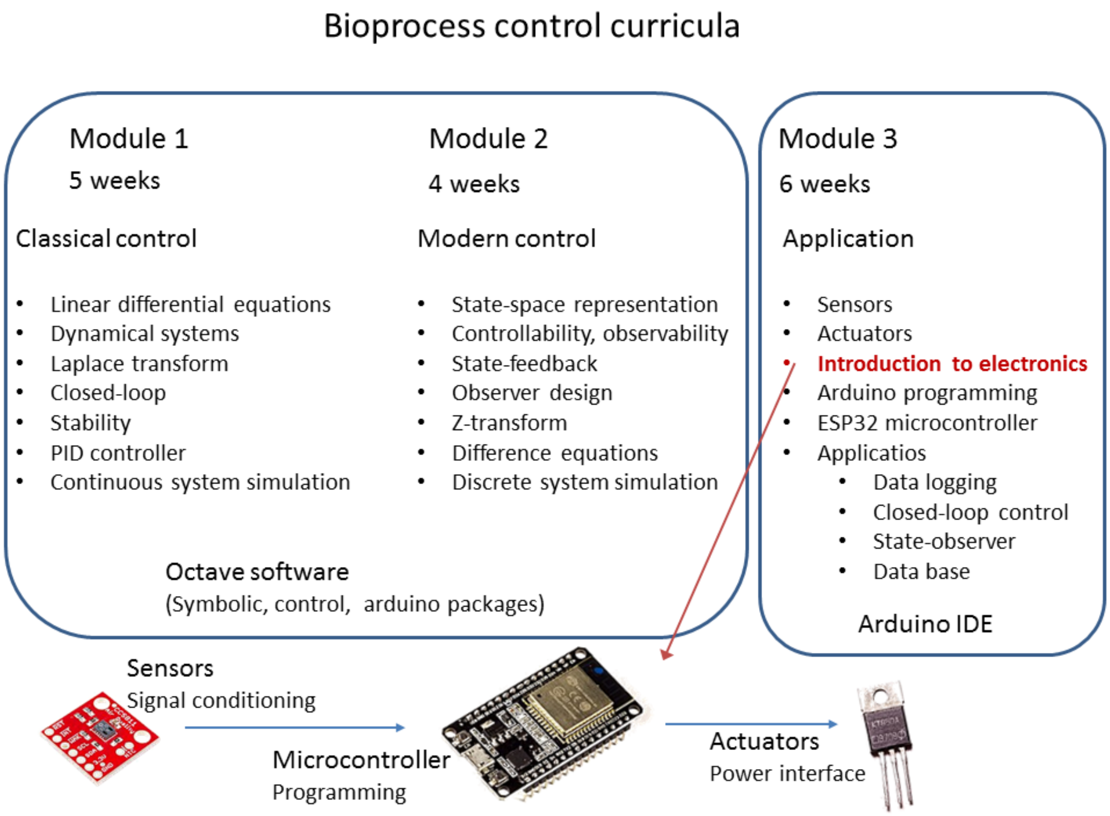

Fig. 1. Topic Covered in the Bioprocess Control Course at PUP.

Finding different resistor values is straightforward; we propose using capacitors of 10μF and resistors of 7.9KΩ. Classical circuits that include operational amplifiers typically invert the signal polarity. The circuit designed to condition the signal from the SKU: SEN0161 pH meter is illustrated in Fig. 2. Similarly, high-pass filters are commonly employed in biomedical engineering to eliminate the signal offset from electrocardiograms (Sun et al., 2002).

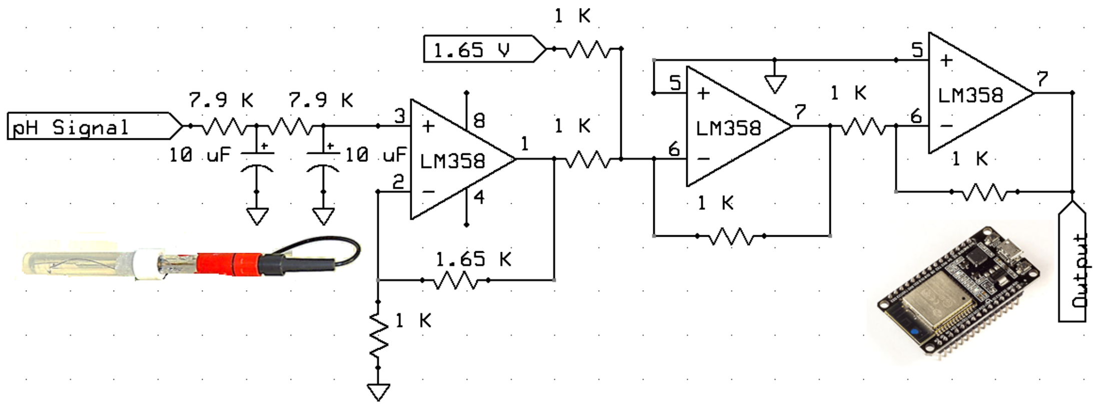

Fig. 2. Signal Conditioning.

The resistors proposed in the signal conditioning have not commercial values, thus a resistors array can be implemented to approximate these values, to avoid confusion in the students, series and parallel connection are explain. In this way, by using only two resistors is possible to obtain an equivalent resistance to be used in the operational amplifier circuit. If they are connected in parallel, the resulting is the inverse of the addition of their inverses (for two resistors is the same than the multiplication over the addition) $R_{eq} = (R1R2)/(R1 + R2)$. The commercial values of resistors and capacitors are multiples of 10, 12, 15, 18, 22, 27, 33, 39, 56, 68 and 82. In this case, the capacitors have 10µF. For the resistor of 1.65KΩ, we can use two resistors of 1.8KΩ and 22KΩ in parallel to obtain an equivalent resistor of 1.66KΩ. For the low-pass filter were proposed resistors of 7.9KΩ and can be obtained connecting in parallel resistors of 8.2KΩ and 220KΩ to have Req = 7.9KΩ.

A similar signal conditioning circuit can be designed for the LM35 temperature sensor, which outputs 10mV for each degree Celsius and has a precision of ±0.5◦C. The MQ2 is an affordable sensor capable of detecting butane, propane, methane, alcohol, and hydrogen, i.e., fuel gases, in concentrations ranging from 300 to 10,000 ppm. This sensor has two outputs: one analog and one digital. The digital output provides a binary signal (1 or 0) indicating whether the fuel gas has been detected, while the analog signal can be read by a computer or microcontroller and is proportional to the concentration of the fuel gas (Babu et al., 2024).

On the other hand, some sensors communicate measurements using communication protocols like I^2C or SPI. For example, in chemical and environmental engineering, the DHT11 is used to measure temperature and relative humidity (Hasan, 2023). The DHT11 can measure temperatures between [0◦C, 50◦C] with a precision of ± 2◦C and a resolution of 1◦C, while humidity can be measured from 20% RH to 90% RH using the 1-wire protocol.

The HDC1080 temperature sensor utilizes the I^2C protocol, allowing the connection of up to 1008 sensors or slave microcontrollers to a master device (Lynch et al., 2016). For measuring oxygen levels using a computer or microcontroller, we recommend the SEN0237-A sensor, which can measure oxygen levels ranging from [0 mg/l to 20 mg/l] and provides an analog signal between [0V and 3V]. This signal can be conditioned for use in a microcontroller's analog input. Mahmoud et al. (2023) developed an aquaponic system utilizing a Raspberry Pi for data acquisition and control along with the aforementioned dissolved oxygen sensor.

The concentration of CO_2 can be measured using the CCS811 sensor, which detects CO_2 levels in the atmosphere and reports the value in ppm via I^2C. This sensor also measures volatile organic compounds (VOCs) such as aldehydes, ketones, alcohols, amines, and carbon monoxide (Anwar et al., 2022). It can detect VOC concentrations ranging from 0 to 32,768 ppm and CO_2 concentrations from 400 to 29,206 ppm.

2.1 Analog Read

When a non-digital sensor is used and the microcontroller reads the signal, an integer value is obtained from the analog-to-digital converter. The control algorithm, including state observers, typically utilizes floating-point values, so a conversion must be implemented to obtain the actual measured value. The analog-to-digital converter of the ESP32 microcontroller has a resolution of 12 bits. In this case, a voltage input ranging from [0V to 3.3V] produces an integer value in the microcontroller between [0 and 212 − 1=4095]. Consequently, a floating-point value can be obtained using a casting or by performing an operation that is saved into a float variable.

Signal conditioning can involve either an offset or amplification of the voltage read by the microcontroller. Performing the inverse operation to obtain the approximate value is straightforward. For example, in the case of the pH meter described above, the read voltage is between [0V and 3.3V]. If the pH is 4, the sensor outputs 236.64 mV. The signal conditioning amplifies this value by a factor of 3.98 and then adds 1.65 V. Thus, the voltage at the analog input becomes 2.59 V, and an integer variable called int ReadpH will hold the value 3214. To derive the pH value from this reading, it is necessary to store the value in a floating-point variable and perform the inverse operations as follows: `float pHvalue = (ReadpH * 3.3 / 4095 - 1.65) / 3.98`, resulting in a real sensor voltage of pHvalue = 236.19 mV. However, in this case, the sensor outputs -414.12 mV for a pH of 14 and 414.12 mV for a pH of 0. Therefore, to obtain the pH value, it is required to compute:

$$\text{FinalpH} = 7 - 14\left(1 - \frac{\text{pHvalue} + 414.12\text{mV}}{2(414.12\text{mV})}\right) = 3.99. \quad \textbf{(3)}$$

Digital sensors transmit values via a communication protocol, in which case the data are read as text (string variables in programming). If computations with this value are required, a function must be created to convert the string variable into a floating-point variable. By using the function `string.toFloat()`, it is possible to obtain a float value.

2.2 Digital Communications

Digital sensors utilize different types of communication protocols. For biotechnological engineering students, it is not practical to explain each protocol; at the PUP, it is only mentioned that there are synchronous and asynchronous methods for communication between devices, and the number of wires used in each case varies. We recommend using sensors that transmit data via I2C.

A general overview of the differences between three communication protocols is presented in Fig. 3 and Table 1. While they are additional communication protocols used in industry, the devices that employ them are often expensive and difficult for universities in developing countries to acquire, or they are not open access, which makes it necessary to resort to costly platforms.

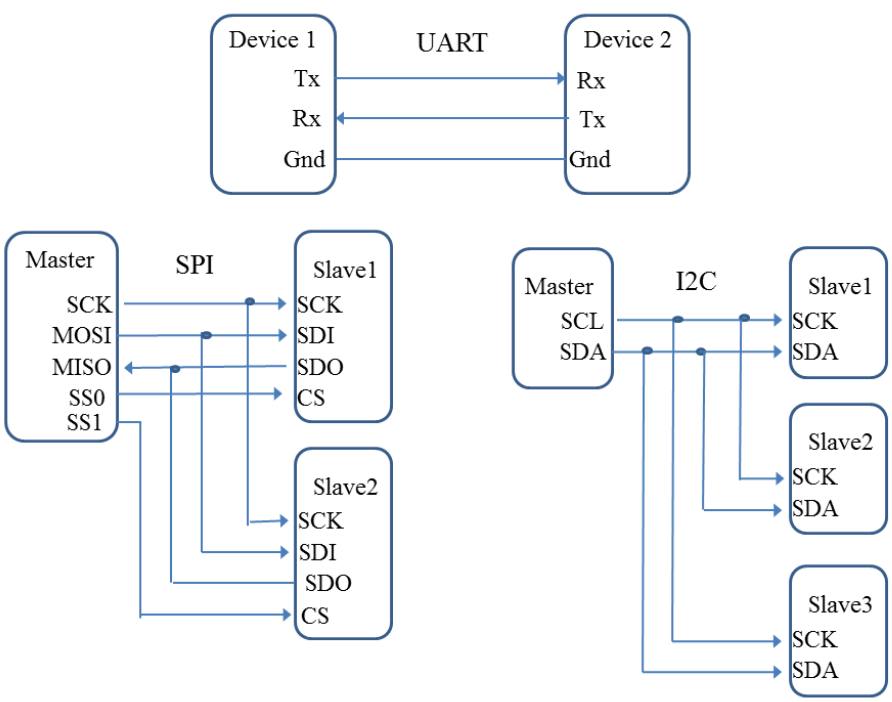

Fig. 3. Communication Protocols Scheme.

Table 1. Communication Protocols Characteristics

Protocol Characteristic	UART	SPI	I^2C
Velocity	469 Kb/s	20 Mb/s	3.4 Mb/s
Master/Slaves	No masters	1 Master, Multiple Slaves	Multiple Masters, Multiple Slaves
No. of Wires required	2	4	2
Duplex	Full Duplex	Full Duplex	Half Duplex

3 Power Interface for Actuators

Actuators are responsible for providing energy and supplying inputs or nutrients to the process. Actuators can include heaters, coolers, mixers, pumps, etc., each of which may operate at different voltages and require different types of signals. Typically, actuators operate at $12V_{DC}$ or $220V_{AC}$. Power regulation can be achieved by controlling the DC voltage; however, managing the electrical current intensity is challenging because computers and microcontrollers typically provide only 20mA. Therefore, a power interface is necessary to supply energy to the actuators.

To control actuators that require $12V_{DC}$, one option is to use a transistor. There are several types of transistors, including bipolar junction transistors (BJTs), insulated gate bipolar transistors (IGBTs), and metal-oxide semiconductor field-effect transistors (MOSFETs). The MOSFET is recommended because it has an isolated gate, allowing the transistor to function as a voltage source controlled by a voltage input, meaning it does not require input current (Wu et al., 2024). If a BJT is used, a cost-effective option is the Darlington TIP120, which can supply 5A while operating at 55V. The input current at its base must be at least 5mA due to its gain hFE ≈ 1000.

For larger loads, non-Darlington BJT transistors may be used; for example, the TIP3055 can operate at 15A output, supporting 70V. However, its gain is hFE ≈ 20, meaning that to supply 15A to the load, this BJT requires 750mA, which cannot be obtained from a digital device. For this reason, we recommend the MOSFET IRFZ44N, which is cheaper than the TIP3055 and can supply 49A, supports 55V, and does not require input current. A schematic of these transistors is shown in Fig. 4.

Some loads, such as heaters or mixers, can be governed by an AC motor or serve as heating systems that operate at $220V_{AC}$. In these cases, a TRIAC can be used; we propose using the MAC16, which can operate up to 800VAC and support a current intensity of 15A. The recommended MOSFET and TRIAC can receive a PWM signal to regulate the power sent to the actuator.

The ESP32 features two digital-to-analog converters (DACs) with an 8-bit resolution on GPIOs 25 and 26, allowing an analog signal to be obtained from the microcontroller, which is not a PWM switching signal. To work with the DAC, the analog output requires an integer value between 0 and 255 due to the 8 bits of resolution ($28 - 1 = 255$). The function to accomplish this can be written, for example, as `contrSig = analogOutput(150);` where contrSig is an integer variable, and 150 is the integer value provided to the DAC. Once the analog signal to be applied to the actuator has been computed and sent, the power interface can be implemented using the power operational amplifier LM675, which is capable of delivering 3A to the load while being fed with ±5 to $\pm 30V_{DC}$ (Batista et al., 2023).

Before explaining how to program an ESP32, students must be familiar with a breadboard to implement the circuits mentioned. In class, the terminals and electronic diagrams for connections are presented; only concepts regarding power and connections are provided. Moreover, by using the mentioned devices, almost no calculations are necessary.

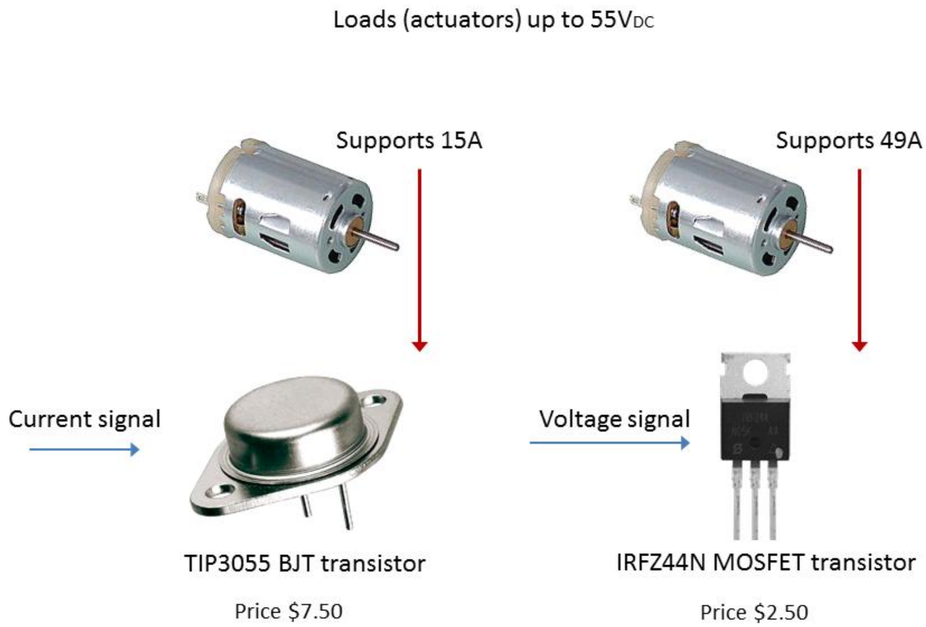

Fig. 4. DC Power Interfaces.

4 ESP32 Programming

Utilizing a microcontroller is an effective approach to teach bioprocess control; they are inexpensive, easy to obtain, and straightforward to program. In the industry, programmable logic controllers (PLCs) are primarily used, but they are costly for students to purchase, especially in developing countries where equipment may be lacking. For instance, the Allen Bradley Micrologix 1000 PLC costs approximately $300.00 dollars. Nevertheless, the ld-micro software can simulate a ladder program in a PIC microcontroller. A guide to PLC program for chemical engineering students was shown by Rowe et al. (2020) where the PLC used is the Allen Bradley Micro830 with a cost of $2294.00 dollars.

A microcontroller is a compact computer containing a microprocessor, memory, peripherals, ADC converters, timers, and more. In particular, a cost-effective and open-source microcontroller was developed by Arduino®. Its simplicity and user-friendly features have made it popular even in high schools for teaching electronics and robotics (Roumen and Fernaeus, 2021). Viana et al. (2016) showed in its Table 1 a comparison among different types of microcontrollers to design underwater sensor nodes.
Recently, the ESP32 microcontroller has emerged as an alternative to the Arduino, being cheaper (around $8.00 dollars) while offering greater computational power. Additionally, it includes Wi-Fi and Bluetooth making it particularly interesting for IoT applications and for students (Márquez-Vera et al., 2023). The ESP32 can be programmed using the Arduino IDE; it is only necessary to add the https://dl.espressif.com/dl/package esp32 index.json URL in the preferences.

The ESP32 has 15 pins that can be utilized to read an analog signal with 12 bits of resolution. For example, to read an analog signal from an LM35 sensor, we can write `#define pin 2` to create a constant named pin equal to 2, and `int Temp;` outside of any function to declare an integer variable named Temp. The Arduino IDE includes the `setup()` function, where we can configure the microcontroller's characteristics. We can then write `pinMode(pin, INPUT);` to define GPIO 2 as an input to receive the analog signal. The `loop()` function is a never-ending cycle where the actions to be executed repeatedly by the microcontroller are programmed. Here, we write `Temp=analogRead(pin);` to obtain the temperature measurement from the LM35.

For digital sensors, certain libraries must be installed. Many sensors are included in Adafruit Industries®; once the libraries are installed, we can write the headers such as `#include <Adafruit_Sensor.h>` to access the sensors. For instance, `#include <Adafruit_BNO055.h>` enables the functions needed to use the BNO055 accelerometer sensor. Additionally, to use devices that communicate via the SPI protocol, it is necessary to add `#include "SPI.h"`. These headers should be written at the beginning of the program.

4.1 Controllers and Observers Computation

In the first two modules of the class, the students work with derivatives and Laplace Transform. However, for real applications using a computer or microcontroller, the dynamical system can be approximated using:

$$\frac{dx(t)}{dt} \approx \frac{x(k+1) - x(k)}{T}, \tag{4}$$

where T is the sample time. In this way, if we have for example

$$\frac{dx(t)}{dt} = a\,x(t), \tag{5}$$

then we can compute (Bastin and Dochain, 1990)

$$x(k+1) = x(k) + aT\,x(k). \tag{6}$$

This approximation changes differential equations for difference equations. To program a controller in a microcontroller is easy by using difference equations, supposing that we have a filter or controller with the Z-transfer function

$$G(z) = \frac{b_0 z + b_1}{z^2 + a_1 z + a_2} = \frac{U(z)}{E(z)}, \tag{7}$$

where $U(z)$ is the control signal and $E(Z)$ the error signal, it is possible to obtain easily the difference equation

$$u(k) = b_0 e(k\text{-}1) + b_1 e(k\text{-}2) - a_1 u(k\text{-}1) - a_2 e(k\text{-}2), \tag{8}$$

being $U(z)$ the Z-transform of discrete time signal $u(k)$, and the same for the error $e(k)$, we can see in (8) that the current control signal to be send by the microcontroller to the power interface $u(k)$ depends on past values of the error and control signal. A more interesting issue is the observer implementation, where state-space representation in used. Taking in consideration (7), its state-space in the controlling form is

$$\begin{gathered} \vec{x}(k+1) = \begin{pmatrix} 0 & 1 \\ -a_2 & -a_1 \end{pmatrix} \vec{x}(k) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(k), \\ y(k) = \begin{pmatrix} b_1 & b_0 \end{pmatrix} \vec{x}(k). \end{gathered} \tag{9}$$

This representation has the general form

$$\begin{gathered} \vec{x}(k+1) = A\vec{x}(k) + Bu(k), \\ y(k) = C\vec{x}(k). \end{gathered} \tag{10}$$

The state space representation has the possibility to feedback the information of all state variables to have a state-feedback to locate the eigenvalues where the designer proposes. However, in biotechnological engineering is not always possible to know all the state variables values in-line. In this case, a state observer must be implemented, for a linear system like (10), the scheme of a state observer is shown in Fig. 5, where the observer gain L must be computed to locate the eigenvalues of ($zI - A+LC$) where they were proposed. Observer design is an important issue in biotechnological engineering due to the lack of in-line sensors (Rocha-Cozatl et al., 2015).

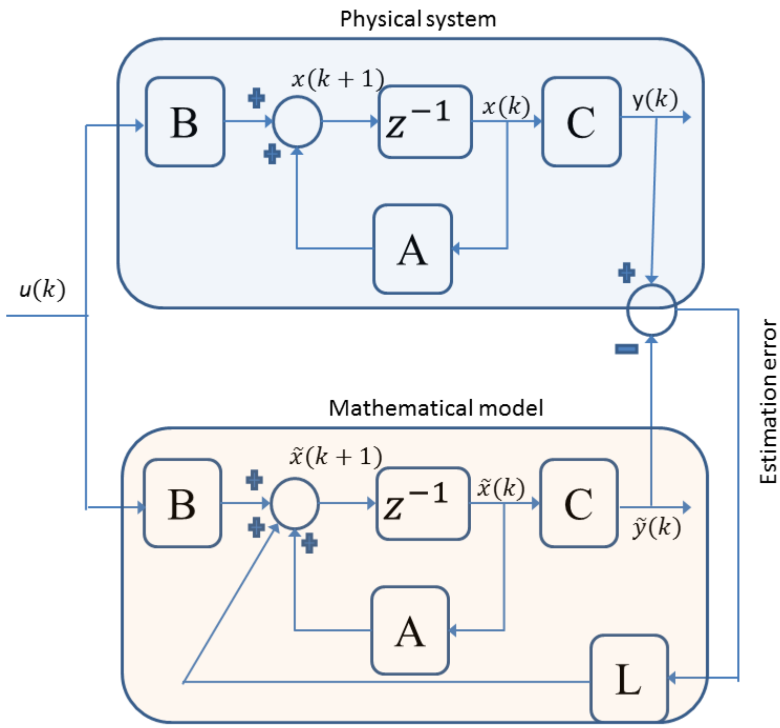

Fig. 5. State Observer.

Similarly to (8), the estimated state $\ddot{x}(k)$ depends on its last valued, the input and the estimation error. Thus, to compute the state observer in the microcontroller we can write outside of the `loop()` function float `xe1_0=0, xe2_2=0, ye=0;` (which is the estimated output) to initialize the past values of the estimated variables and the estimated output. The system output y can be obtained by using the analog input. The estimation error e is the difference between the system output y less the estimated output y_e.

The, once the observer gain $L=(L_1\ L_2)$T was computed, it is possible to write in the program: `xe1 = xe2_0+L1*e;` and for the other variable `xe2 = a2+xe1_0+a1*xe2_0+u+L2*e;` being u the input sent to the system. Finally, the estimated variables must be updated writing `xe1_0=xe1; and xe2_0=xe2;` after the variables, output and error computation.

4.2 Data Logging

The ESP32 has internal memory, but it is possible to use a SD card memory to log the measures obtained by the microcontroller. However, it depends on a good Wi-Fi connection in the classroom. The SD card uses the SPI protocol to communicate with the ESP32. A good explanation that shows hot to connect a micro SD card and the functions used to open, write, append and close a text file is shown in https://randomnerdtutorials.com/esp32-microsd-card-arduino/

To use the SD card and to make files is necessary to write the headers `#include "SD.h"` and `#include "FS.h"`. As was shown in Figure 3 to use the SPI protocol a pin CS must be defined, for example by using the GPIO 5, we can write `#define SD_CS 5` and then in the `setup()` function we write `SPI.begin(SCK,MISO,MOSI,SD_CS);` where the wires SCK (clock), MISO (multiple input single output) and MOSI (multiple output single input) must be defined like the SD CS. The ESP32 has two SPIs, the first one uses the GPIOs 18, 19, 23 and 5 for the VSPI and the GPIOs 14, 12, 13 and 15 for the HSPI.

To validate the SD card connection we write `SD.begin(SD_CS);` in this way, we can see the SD card type and its free memory. Hereby, it is possible to write a file in the SD card by using the functions provided in the url given above.

4.3 Data Base

The ESP32 has Wi-Fi and Bluetooth modules that can be very interesting for the students because they can acquire data and to control remotely the ESP32. In the process control course the Wi-Fi is used to have a data-base where is possible to log the data in addition to the SD card. The Wi-Fi needs the header `#include "WiFi.h"` and the host and key can be saved like definitions using `#define WIFI_SSID "MyNet"` and `#define WIFI_PASSWORD "MyKey"`. Then in the `setup()` function we write `WiFi.begin(WIFI_SSID, WIFI_PASSWORD);`

Once the Wi-Fi connection was made is possible to use a data-base (DB) to have an IoT application. The Firebase is a free DB of Google® (Sung et al., 2023). To access to the DB in required to have an account in Google® where we can create the DB, to assigning a name and even to get a password, to access to the DB from the ESP32 we can write `#define FIREBASE_HOST https://MyDataBaseName` and `#define FIREBASE_AUTH "keyGiveByFirebase"` in the program.

In the setup()function we write `Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);` to connect the ESP32 to the DB `Firebase.set String("variable1", "variable2");` and to define the variables or fields names. Now with the DB identified is possible to get or put values from or to the ESP32 and the DB. The functions used to put a value or file are `setInt(); setFloat(); setString(); setJSON();` and `setFile();` To read from the DB there exist the functions `getInt(); getFloat; getString(); getJSON(); getFile();` In these cases, the value or string is updated deleting the past value. To make the DB saving the measures obtained by the ESP32 is possible to use the function `pushString();`

In this way, students can interact with the ESP32 using their smartphones to retrieve or send information. This activity was well received and enjoyed by the students. For control applications, the reference signal can be set by a user via Firebase.

5 Recommendations and Students Feedback

At the beginning, the students were skeptical about learning electronics. We recommend avoiding the use of theoretical descriptions and complex exercises. Most students at the PUP are pragmatists, so incorporating small practical exercises using electronic devices helps them grasp the concepts better. Assigning online tutorials as homework can also motivate students and help them resolve doubts. The first two modules involve the use of Octave®, which allows for the introduction of the ESP32 as a data acquisition card.

In the last module, they have a basic understanding of programming, sensors, and actuators, which sets the stage for introducing instrumentation and electronics through small applications. There is a wealth of information on the ESP32 available online, and we recommend the resource mentioned in the previous subsection Data logging.
Starting with Arduino programming can be challenging for some students, so to minimize costs, students work in groups of five. This approach allows them to see how an electronic card can perform computations originally done on a PC. Most of the time is spent on programming how to manage the electronic devices, and data logging using an SD card is particularly valuable for students.

We recommend introducing the Wi-Fi and Bluetooth modules of the ESP32 to motivate students to program the ESP32 and interact with a chemical reactor or, at least, with sensors and actuators via the internet. In the May-August 2024 term, two groups of 18 and 23 students respectively from the Biotechnology Engineering program at PUP were taught electronics topics in the final module. It was necessary to demonstrate the use of a breadboard to build circuits. Additionally, the multimeter and voltage readings from analog sensors were shown.
Once the students were able to build small circuits and measure signals using the ESP32 as a data acquisition card, communication protocols were introduced. Students worked with digital sensors and used the ESP32 as a controller. At this stage, the SD card was taught, and practical work involving closed-loop control or automation was proposed.

Finally, a database was created to store and view measurements online, allowing students to control actuators through their smartphones, feature they particularly enjoyed. This final module includes an evaluation component to assess both individual student performance and group project outcomes within competency-based education framework, aimed at fostering meaningful learning (Castelló et al., 2023).

Rubrics were used to evaluate student performance. The rubric includes items such as degree of participation, understanding of the electronic circuit and Arduino programming, ability to answer evaluation questions, use of technical language, and team integration. Each item has four levels of achievement. For example, in the first item, the highest level might be "The student can describe each part of the program and understands the hardware connections on the breadboard." A second level might be "The student has a good understanding of Arduino programming but less so of the electronic circuit, or can correctly describe the hardware connection but does not fully understand the program." A third level might be "The student only understands certain parts of the program and circuit," while the lowest level might be "The student does not understand the program and has little knowledge of electronics."

Similarly, a rubric was created to evaluate the project outcomes. This rubric assesses the quality of the report, the electronic diagram, the calculations made to design the circuit, the program (including the use of functions and indentation), the results obtained, and the use of Wi-Fi or Bluetooth connections. Each item is also evaluated on a four-level scale. For example, if an item is worth 20%, the levels could assign values of 20%, 15%, 10%, or 5% depending on whether the requirement was fully met, needs some corrections, was partially met, or showed no evidence, respectively.

5.1 Student's Comments

Many students initially found control theory concepts difficult to grasp. However, as time passed, they became more comfortable with programming in Octave®. To reinforce the ideas presented in class, especially when equipment was scarce or insufficient, the use of affordable sensors and actuators in practical applications helped illustrate the key concepts.

Students found the interaction with electronics particularly engaging when they could save data to an SD card. While achieving full bioprocess control requires more advanced topics such as nonlinear control and larger project development, which is challenging within the six-week course dedicated to electronics, concepts like automation, data logging, and signal conditioning provided practical applications of the theoretical concepts discussed in class.

Students especially enjoyed working with Wi-Fi applications, where they could save data to the cloud, remotely control actuators, and collaborate with other groups to complete larger projects by sharing information in real-time.

At the end of the course, students were interviewed about the teaching methods, the difficulty of each module, their satisfaction with the topics covered, and their perceived level of expertise in achieving the educational objectives. Overall, the two groups provided positive evaluations and expressed satisfaction with the course. The primary challenges reported were related to understanding the theoretical concepts introduced in the first module. The level of acceptance was 87%, around 9% had doubts about the importance of automation and electronics for biotechnology engineering, whereas the remaining 4% was indifferent.

5.2 Evaluation Process

The lowest grades were typically obtained in the first module, where control theory is introduced. This is primarily due to deficiencies in mathematics, despite students having previously studied topics such as differential equations, Laplace transforms, and numerical methods. Once the theoretical framework was established, grades improved, particularly when students engaged in practical projects. One of the main challenges in evaluation is assessing each individual within a group, as some students tend to participate less, or there are issues with teamwork and organization. Some students prefer to focus on working with the computer, while others prefer hands-on work with electronics.

The use of rubrics has proven helpful for evaluating both individual students and groups. Additionally, rubrics allow students to understand the level of mastery required for each topic to achieve a high grade. In Mexico, the grading scale typically ranges from 0 to 10, and at PUP, students must achieve at least a 7 to pass the course. The average group grades were 8.7 and 8.9, respectively, with the breakdown in the first group as follows: (7.5 + 9.4 + 9.3) = 8.7, where five students did not pass the first module, whereas in the second group only two students did not pass the first module.

6 Conclusions

The teaching of electronics and process control in biotechnological engineering program is essential for preparing students to meet the challenges of Industry 4.0. However, economic limitations and the lack of access to specialized software in developing countries can hinder the acquisition of advanced digital competencies. In this context, the use of affordable microcontrollers like the ESP32 and open-source software tools has proven to be a viable and effective solution in training future engineers.

This article has demonstrated how the design of data acquisition systems, signal conditioning, and the implementation of power interfaces can be taught using low-cost platforms without compromising educational quality.

In terms of learning assessment, students were found to face greater challenges in understanding theoretical concepts, particularly in control and advanced mathematics. However, by introducing hands-on projects, their performance improved significantly. This highlights the importance of combining theory with practical applications to ensure a better grasp of complex topics. The use of detailed rubrics has facilitated both individual and group assessments, providing clearer feedback on areas needing reinforcement.

In conclusion, it is possible to successfully implement the teaching of process control and electronics using accessible and affordable resources, enabling educational institutions to overcome financial barriers and provide students with robust and relevant training.

7 References

Ahmad, R., Choudhary, N., Gupta, S. K., Joshi, A. M., & Boolchandani, D. (2023). Novel tunable current feedback instrumentation amplifier based on BBFC op-amp for biomedical applications with low power and high CMRR. *Integration*, *90*, 214-223. https://doi.org/10.1016/j.vlsi.2023.02.003

Anwar, O., Keating, A., Cardell-Oliver, R., Datta, A., & Putrino, G. (2022). Design and development of low-power, long-range data acquisition system for beehives – BeeDAS. *Computers and Electronics in Agriculture*, *201*, Article 107281. https://doi.org/10.1016/j.compag.2022.107281

Babu, T., Nair, R. R., Kishore, S., & Vineeth, M. (2024). Enhancing gas leak detection with IoT technology: An innovative approach. *Procedia Computer Science*, *235*, 961-969. https://doi.org/10.1016/j.procs.2024.04.091

Barry, B. I. A. (2009, December). Using open source software in education in developing countries: The Sudan as an example. In *Proceedings of the International Conference on Computational Intelligence and Software Engineering (CISE)* (pp. 1-4). IEEE. https://doi.org/10.1109/CISE.2009.5364872

Bastin, G., & Dochain, D. (1990). *On-line estimation and adaptive control of bioreactors*. Elsevier.

Batista, D. S., Granziera-Jr, F., Tosin, M. C., & de Melo, L. F. (2023). Analysis and practical implementation of a high-power Howland current source. *Measurement*, *207*, 112404. https://doi.org/10.1016/j.measurement.2022.112404

Bertsatos, A., & Chovalopoulou, M. E. (2019). A novel method for analyzing long bone diaphyseal cross-sectional geometry: A GNU Octave CSG toolkit. *Forensic Science International*, *297*, 65-71. https://doi.org/10.1016/j.forsciint.2019.01.041

Botero, M. L., Selmer, A., Watson, R., Bansal, M., & Kraft, M. (2016). Cambridge WebLabs: A process control system using industrial standard Simatic PCS 7. *Education for Chemical Engineers*, *16*, 1-8. https://doi.org/10.1016/j.ece.2016.04.001

Caccavale, F., Gargalo, C. L., Gernaey, K. V., & Krühne, U. (2023). *SPyCE: A structured and tailored series of Python courses for (bio)chemical engineers*. *Education for Chemical Engineers*, 45, 90-103. https://doi.org/10.1016/j.ece.2023.08.003

Caro, F., Ferreira, J., Castelló, E., Pinto, J. C., & Santiviago, C. (2024). *Bio2Py: An API for integrating Python with BioWin for enhanced data acquisition in wastewater treatment simulations*. *Journal of Water Process Engineering*, 63, 105426. https://doi.org/10.1016/j.jwpe.2024.105426

Castelló, E., Santiviago, C., Ferreira, J., Coniglio, R., Larnaudie, E. B. V., Passeggi, M., & López, I. (2023). *Towards competency-based education in the chemical engineering undergraduate program in Uruguay: Three examples of integrating essential skills*. *Education for Chemical Engineers*, 44, 54-62. https://doi.org/10.1016/j.ece.2023.05.004

Cugnet, M., Gallois, F., Kirchev, A., & Dutykh, D. (2023). *Neolab: A Scilab tool to simulate the negative electrode of lead-acid batteries*. *SoftwareX*, 22, 101394.

Domínguez, J., Alonso, M., González, E., Guijarro, M., Miranda, R., Oliet, M., Rigual, V., Toledo, J., Vallar-Chavero, M., & Yustos, P. (2021). *Teaching chemical engineering using Jupyter Notebook: Problem generators and lecturing tools*. *Education for Chemical Engineers*, 37, 1-10. https://doi.org/10.1016/j.ece.2021.06.004

Faritha, B., Revathi, R., Suganya, M., & Gladiss, M. (2020). *IoT based cloud integrated smart classroom for smart and a sustainable campus*. *Procedia Computer Science*, 172, 77-81. https://doi.org/10.1016/j.procs.2020.05.012

Feise, H., & Schaer, E. (2021). *Mastering digitized chemical engineering*. *Education for Chemical Engineers*, 34, 78-86. https://doi.org/10.1016/j.ece.2020.11.011

Gunasekera, M., Ahmed, S., & Khan, F. (2020). *Integration of process safety in equipment design: A framework for academic learning activity*. *Education for Chemical Engineers*, 30, 32-39. https://doi.org/10.1016/j.ece.2019.09.004

Hasan, M. W. (2023). *Building an IoT temperature and humidity forecasting model based on long short-term memory (LSTM) with improved whale optimization algorithm*. Memories - Materials, Devices, Circuits and Systems, 6, 100086. DOI: 10.1016/j.memori.2023.100086

Hrnčiřík, P., & Kohout, J. (2024). *A Matlab-based simulator for the study of process control of fed-batch yeast fermentations*. Education for Chemical Engineers, 49, 67–77. DOI: 10.1016/j.ece.2024.06.001

Islam, D. T., Williams, M. R., Teppen, B. J., Johnston, C. T., Li, H., Boyd, S. A., Zylstra, G. J., Fennell, D. E., Cupples, A. M., & Hashsham, S. A. (2024). *Comprehensive model for predicting toxic equivalents (TEQ) reduction due to dechlorination of PCDD/F congeners*. Journal of Hazardous Materials, 480, 135749. DOI: 10.1016/j.jhazmat.2024.135749

Jhurree, V. (2005). *Technology integration in education in developing countries: Guidelines to policy makers*. International Education Journal, 6(4), 467–483.

Kakkar, S., Kwapinski, W., Howard, C., & Kumar, K. (2021). *Deep neural networks in chemical engineering classrooms to accurately model adsorption equilibrium data*. Education for Chemical Engineers, 36, 115–127. DOI: 10.1016/j.ece.2021.04.003

Khan, F., Amyotte, P., & Adedigba, S. (2021). *Process safety concerns in process system digitalization*. Education for Chemical Engineers, 34, 33–46. DOI: 10.1016/j.ece.2020.11.002

Lynch, K. M., Marchuk, N., & Elwin, M. L. (2016). Chapter 13 - I2C communication. In *Embedded Computing and Mechatronics with the PIC32* (pp. 191–211). Newnes (Elsevier).

Mahmoud, M. M. M., Darwish, R. R., & Bassiuny, A. M. (2024). *Development of an economic smart aquaponic system based on IoT*. Journal of Engineering Research, 12(4), 886–894. DOI: 10.1016/j.jer.2023.08.024

Márquez-Vera, M. A., Martínez-Quezada, M., Calderón-Suárez, R., Rodríguez, A., & Ortega-Mendoza, R. M. (2023). *Microcontrollers programming for control and automation in undergraduate biotechnology engineering education*. Digital Chemical Engineering, 9, 100122. DOI: 10.1016/j.dche.2023.100122

Ozdilek, E. E., Ozcakar, E., Muhtaroglu, N., Simsek, U., Gulcan, O. & Sendur, G. K. (2024). A finite element based homogenization code in Python: Hompy. Advances in Engineering Software 194, 103674. https://doi.org/10.1016/j.advengsoft.2024.103674

Paoli, L. T., Inguva, P. K., Haslam, A. J. & Walker, P. J. (2024). Confronting the thermodynamics knowledge gap: A short course on computational thermodynamics in Julia. Education for Chemical Engineers 48, 1–14. https://doi.org/10.1016/ j.ece.2024.03.002

Quive, L. G., Leandro, S., Bandali, E. C., Gueze, G. A., João, D. A., Gomundanhe, A. M., Neuana, N. F. & Macuvele, D. L. P. (2021). Exploring materials locally available to teach chemistry experimentally in developing countries. Education for Chemical Engineers 34, 1–8. https://doi.org/10.1016/j.ece.2020.09.004

Reverter, F. & Areny, R. P. (2009). Circuitos de interfaz directa sensor-microcontrolador. Marcombo, Barcelona, Spain.

Rocha-Cozatl, E., Sbarciog, M., Dewasme, L., Moreno, J. & Wouwer, A.V. (2015). State and input estimation of an anaerobic digestion reactor using a continuous-discrete unknown input observer. IFAC-PapersOnLine 48 (8), 129–134. https://doi.org/10.1016/j.ifacol.2015.08.169

Roumen, G. J. & Fernaeus, Y. (2021). Envisioning Arduino action: A collaborative tool for physical computing in educational settings. International Journal of Child-Computer Interaction 29, 100277. https://doi.org/10.1016/j.ijcci.2021.100277

Rowe, S. C., Samson, C. I. & Clough, D. E. (2020). A framework to guide the instruction of industrial programmable logic controllers in undergraduate engineering education. Education for Chemical Engineers 31, 76–84. https://doi.org/10.1016/j.ece.2020.03.001

Sanbatoro, L., Laveglia, A., Ukraincsyk, N. & Koenders, E. (2023). Life cycle assessment modelling in Octave/Matlab: Hydrated lime manufacturing case study. Materials Today: Proceedings, Article in Press. https://doi.org/10.1016/j.matpr.2023.08.002

Santos, G. F. & Reis, F. B. (2021). Automated analytical procedure using multicommuted flow analysis and organic solvent extraction controlled by an Arduino due board for photometric determination of zinc in water. Microchemical Journal 163, 105918. https://doi.org/10.1016/j.microc.2021.105918

Saravanan, G., Parkhe, S., Thakar, C., Kulkarni, V., Mishra, H. & Gulothungan, G. (2022). Implementation of IoT in production and manufacturing: An industry 4.0 approach. Materials Today: Proceedings 51 (8), 2427–2430. https://doi.org/10.1016/j.matpr.2021.11.604

Seifrid, M., Strieth-Kalthoff, F., Haddadnia, M., Wu, T. C., Alca, E., Bodo, L., Arellano-Rubach, S., Yoshikawa, N., Skreta, M., Keunen, R. & Aspuru-Guzik, A. (2024). Chemspyd: An open-source Python interface for chemspeed robotic chemistry and materials platforms. Digital Discovery 3 (7), 1319–1326. http://doi.org/10.1039/d4dd00046c

Sun, Y., Chan, K. L. & Krishnan, S. M. (2002). ECG signal conditioning by morphological filtering. Computers in Biology and Medicine 32 (6), 465–479. https://doi.org/10.1016/S0010-4825(02)00034-3

Sung, W. T., Isa, I. G. T. & Hsiao, S. J. (2023). An IoT-based aquaculture monitoring system using Firebase. Computers, Materials and Continua 76 (2), 2179–2200. https://doi.org/10.32604/cmc.2023.041022

Thapa, P. & Gautam, A. (October 2021). Potential of free and open source software for education in developing countries. In: FOSS4G-Asia. International Association of Online Engineering, Dhulikhel, Kavrepalanchok, Nepal, pp. 168–172.

Viana, S. S., Vieira, L. F. M., Nacif, J. A. M. & Vieira, A. B. (2016). Survey on the design of underwater sensor nodes. Design Automation for Embedded Systems 20, 171–190. https://doi.org/10.1007/s10617-015-9169-6

Vishwanatha, J., Swamy, R. S., Mahesh, G. & Gouda, H. V. (2023). A toolkit for computational fluid dynamics using spectral element method in Scilab. Materials Today: Proceedings 92 (1), 209–214. https://doi.org/10.1016/j.matpr.2023.04.341

Wu, X. D., Wang, Y., Yu, C. H., Fei, X. X., Yang, J. Q. & Li, X. J. (2024). Impact of SiC power MOSFET interface trap charges on UIS reliability under single pulse. Microelectronics Reliability 155, 115375. https://doi.org/10.1016/j.microrel.2024.115375

Zárate-Navarro, M. A., Schiavone-Valdez, S. D., Cuevas, J. E., Warren-Vega, W. M., Campos-Rodríguez, A. & Romero-Cano, L. A. (2024). Stem activities for heat transfer learning: Integrating simulation, mathematical modeling, and experimental validation in transport phenomena education. Education for Chemical Engineers 49, 81–90. https://doi.org/10.1016/j.ece.2024.06.004