# A Project Based Learning Approach for development of an experimental setup and a simulator for position and velocity control of a DC motor with interactive and pre-calculated parameters.

**Prof. Fernando Silveira Madani, Mauá Institute of Technology**

Fernando Silveira Madani received the B.S (1998) in Mechatronics Engineering from the Univ. Paulista – Brazil, the M.S. (2002) and Ph.D. (2010) from the Aeronautical Institute of Technology (ITA) - Brazil. In 2002, he joined the faculty of the Dept. of Mechanical Engineering, Mauá Institute of Technology – Brazil, where he is currently as a full professor and Head of the Mechatronics Engineering program. His main research interests include, robotics, advanced manufacturing systems, embedded systems, and autonomous mobile robots. Since 2014 is an INEP (agency linked to the Ministry of Education) advisor, to promote the evaluation and improvement of undergraduate courses in mechatronics engineering in Brazil.

**Mrs. Andressa Corrente Martins, Instituto Maua de Tecnologia**

Andressa Martins is holds a master's degree in Aerospace Systems and Mechatronics with a focus on Robotics from the Aeronautics Institute of Technology and a degree in Control and Automation Engineering from the Universidade Paulista. Currently, she is a professor at the Mauá Institute of Technology. She has experience in the field of Control and Automation Engineering (Mechatronics), working mainly on the following topics: Intelligent Systems, Digital Twin, Industrial Automation, Instrumentation, Robotics, Mechanism Design, and Engineering Fundamentals.

**Leonardo Oneda Galvani, Instituto Maua de Tecnologia**

Student of control and automation engineering at the Maua Institute of Technology.

**Dr. Anderson Harayashiki Moreira, Instituto Mauá de Tecnologia**

Graduated in Control and Automation Engineering from Instituto Mauá de Tecnologia (IMT) (2008). Master in Mechatronics Engineering from the Instituto Tecnológico de Aeronáutica (ITA) (2011). PhD in Mechatronics Engineering from the Instituto Tecnológico de Aeronáutica (ITA) (2017). He is currently a professor at the Instituto Mauá de Tecnologia. He develops activities and research in the area of mobile autonomous robotics, control systems, industrial robotics and microcontroller systems.

**Prof. Alexandre Harayashiki Moreira M.S., Instituto Mauá de Tecnologia**

Alexandre Harayashiki Moreira received M.S. (2017) in multirobot system from UFABC Univ., Brazil. Since 2016, he has been a Member of Control & Automation Research group at Maua Institute of Technology focused on autonomous robots and smart cities.

# A Project Based Learning Approach for development of an experimental setup and a simulator for position and velocity control of a DC motor with interactive and pre-calculated parameters


## ABSTRACT

The project involves developing an experimental setup and an interactive simulator that will control the angular position and velocity of an inertia wheel coupled to a DC motor. The project was developed using the knowledge from the subjects studied in the 4th year of the Control and Automation Engineering course at the Instituto Mauá de Tecnologia, including Object-Oriented Programming, Database, Instrumentation, Embedded Systems, and Control Systems. This work will allow for experimenting and understanding important concepts related to these disciplines. To achieve this, this project will have several phases, including the development of the virtual environment, which will have two stages. The first stage involves interactive control of the gains of a PID controller that will control the position in degrees (°) and the velocity in rotations per minute (RPM). The second stage involves analyzing the system's behavior with pre-designed controllers for controlling only the position in degrees (°). In both stages, there will be the visualization of the simulation results. The second phase involves building the necessary hardware to perform what the simulator requires, including specifying the components and their interconnections, and finally, programming the system (firmware and software-GUI) to operate according to the specifications provided by the simulator user. At the end, examples of projects will be presented as well as their evaluation, and focused on ABET, how these projects can be used to evaluate students' outcomes.


**Keywords:** Project Based Learning, Integrative project, multidisciplinary project, control, instrumentation and simulator.


## INTRODUCTION

The integration of Instrumentation, Microcontrollers and Control Systems I disciplines in this technical project represents a significant milestone in the interdisciplinary approach to engineering, allowing a practical and complete application of the concepts learned throughout the course. A. Ribas Neto, M. Fiorin, and T. Dequigiovani [1] highlight the value of project-based learning in technology degree programs to deepen students' understanding and skills in the field. In this report, the development of an interactive simulator designed to control the speed and position of a motor coupled to an inertia wheel will be presented. This simulator not only provides the opportunity to experience and understand fundamental concepts from these disciplines, but also illustrates how the convergence of hardware and software can create practical and educational solutions.

This project consists of developing an interactive simulator that will control the angular position and speed of an inertia wheel coupled to a DC motor, because according to [2] DC motor is one of the widest actuators utilized in various control applications. The project was developed using knowledge from the subjects studied in the 4th Series of the control and automation engineering course at the Mauá Institute of Technology, namely: Object Oriented Programming and Database, Instrumentation,

Microcontrollers and Control Systems. This simulator will allow you to experience and understand important concepts related to these disciplines. Undoubtedly, the importance and stature of the field of control are evidenced by the number of annual national and international meetings and conferences, publications (including textbooks) and, more importantly perhaps, its impact on industrial applications that touch the lives of everyone [3].

This simulator has the same objectives as mentioned [4]:

- Demonstrating/validating/motivating analytic concepts.
- Introducing real world control/modeling issues, such as saturation, noise, sensor/actuator dynamics, uncertainty, etc.
- Providing facility with instrumentation and measurement tools.
- Exposing students to broader design issues from problem specification to hardware implementation and economic considerations.
- Exposing students to professional practice that includes maintaining engineering notebooks and report writing.
- Team learning and problem solving.
- Comparing theoretical results with real world results, thus validating the theory

To achieve this, there will be some phases of this project, namely: the development of the virtual environment, which will include two stages. The first part consists of the interactive control of the gains of a PID controller that will control the position in degrees (°) and the speed in revolutions per minute (RPM). In the second stage, there will be an analysis of the system's behavior with pre-designed controllers to control only the position in degrees (°), and in both stages there will be visualization of the simulation results. The second phase of the project consists of building the necessary hardware to perform what is required by the simulator, involving the specification of the components and the structure between them, and, finally, the programming for the system to function in accordance with what is specified by the simulator user.

## *GOALS*

a. Develop a simulation environment for different closed control loops on the same device.

There are several types of controllers and controllers tuning methods, as you can see in [5], but this project will address PID variations and a phase advance for greater understanding.

The project aims to develop an effective simulation platform by allowing interactivity and adjustment of PID controller gains, and by allowing comparison of these with other types of controllers, through controlling both the position and speed of an inertia wheel coupled to a DC motor.

This approach enables direct comparison between different control strategies, providing information about the specific effects of parameters such as proportional, integral, and derivative gain on system responses. This comparative analysis capability contributes to the optimization of desired control, and to the development of a deeper understanding of control principles and their practical application.

b. Practical Application and Interdisciplinary:

The project focuses on the practical application of engineering concepts, to provide a better understanding of theoretical knowledge, in addition to the implementation and testing of important concepts for a professional in the Control and Automation area, such as mathematical identification of systems, PID control and microcontroller programming.

In parallel, the interdisciplinary integration between the areas of Instrumentation, Microcontrollers, Control Systems and Object-Oriented Programming allows for an in-depth understanding of how these areas relate and collaborate in complex engineering projects. This develops critical thinking and develops skills that focus on solving problems and facing real-world demands, where solutions require collaboration between different areas of knowledge.

## *JUSTIFICATION*

The development of the interactive simulator to control angular position and speed is a highly relevant initiative in the context of the control and automation engineering course. This project is a practical response to the knowledge acquired in the 4th Series subjects, allowing students to understand how this knowledge translates into real solutions and promoting a deeper understanding of the concepts and theories learned.

Furthermore, the construction and operation of the simulator requires the development of complex technical skills, such as dealing with sensors, motors, encoders, microcontrollers, programming dynamic system controllers, graphical interfaces for simplified human interaction, among other technical skills developed.

The project structure, divided into progressive phases, reflects common steps in real-world engineering projects. Conceptualizing a system, simulating it, and validating it is an integral part of developing a project, from conception to implementation, emphasizing practical experimentation in a controlled environment.

Finally, the development of this simulator is justified by its educational and typical use benefits, as it is a functional and effective control tool. The simulator has the potential to serve as a resource for students and professionals, allowing for virtual testing and experimentation before implementation in real systems.

## THEORETICAL FOUNDATION

Before entering in the studies of the control theory involved in the project, the functioning of the steps involving the virtual environment, the components present and the logic between them will be discussed.

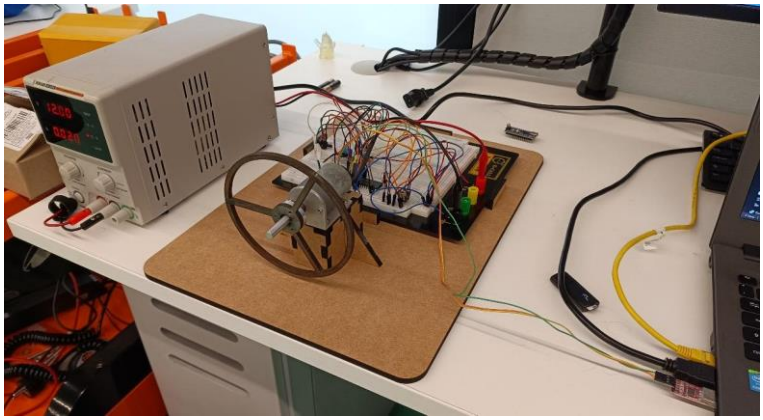
### *HARDWARE*

As discussed previously, the principal component of the project is the DC motor, in which an inertia wheel is coupled to the output shaft of a reduction box. The reason for its inclusion in the system is to create an additional load, providing a greater challenge in controlling the engine. In the same motor assembly, we find the motor rotor coupled to the reduction box input shaft, as well as a Hall effect encoder, as illustrated in the following image:

Figure 1 — DC motor with flywheel.

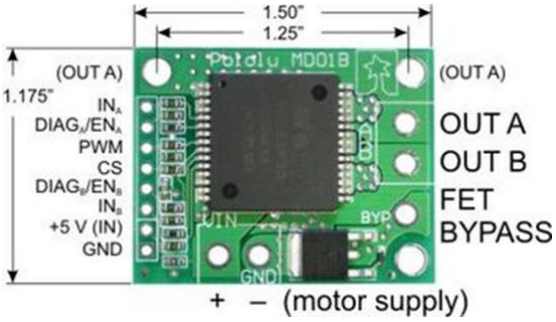
However, in addition to the engine, all the electronics that make the use and control of the system possible, this set is called the plant, with the following image:

Figure 2 — Complete Control Plant



When it comes to control, the first consideration is given to analog quantities. For example, it was necessary to adjust the voltage applied to a motor due to its different speeds. For this, an H bridge is used, which manages the voltage and direction of the motor through two digital signals and a PWM (Pulse Width Modulation). The specific H-bridge configuration used in the project is illustrated below:

Figure 3 — H Bridge




Since digital pins and a PWM are needed to control the voltage and direction of the motor, a microcontroller, the PIC16F18877, was included in the project. It will control the H bridge by sending the data according to the implemented controller. In addition to being connected to the H bridge, the microcontroller will also be responsible for counting the pulses generated in the encoder and identifying, through channels A and B, the direction of rotation of the motor and finally assigning its value correctly.

At this point, the hardware would already be capable of executing the control, but the idea of the project is to create a simulator. To do this, existing hardware must be connected to the computer. Thus, serial communication was used between the microcontroller and the computer.

***INSTRUMENTATION***

For the Instrumentation part of the discipline, in addition to the project proposal, the sensor used was the motor's own hall effect encoder, therefore it was necessary to understand how the motor encoder used works and how it is read.

A Hall effect encoder is a device used to measure the position, speed, and direction of rotation of a shaft in mechanical systems, particularly in electric motors, such as DC (direct current) motors. This type of encoder is based on the hall effect, which is a physical phenomenon in which an electrical voltage is generated in a conductor when it moves through a magnetic field.

In the context of hall effect encoders, there is usually a magnet mounted on the rotating shaft of the motor. These sensors sensitive to magnetic fields are strategically placed around the magnet. When the shaft rotates, changes in the magnetic field are detected, and generates electrical signals proportional to the variations in the magnetic field.

The encoder has two output channels called Channel A and Channel B. The phase relationship between the signals in channels A and B is used to determine the direction of rotation.

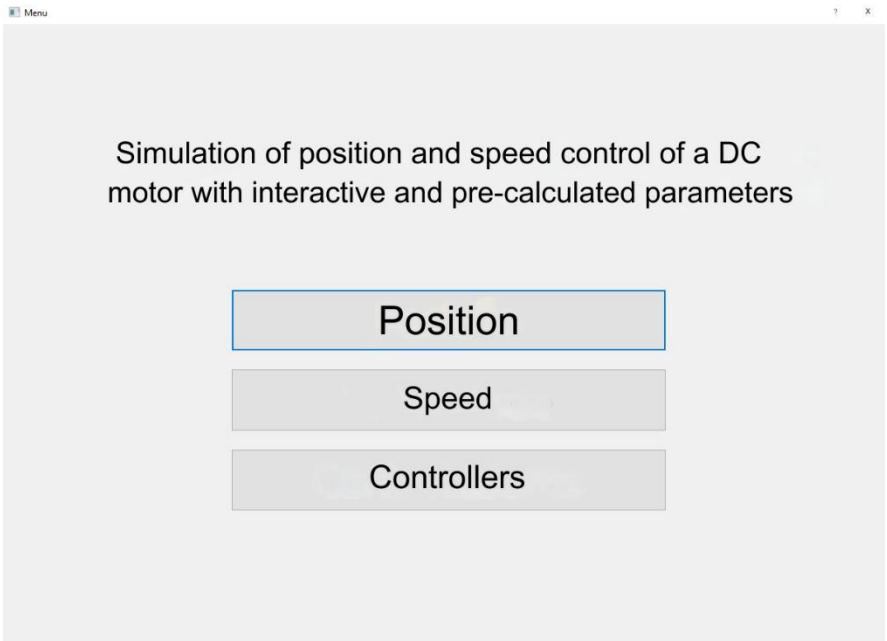
These encoders are essential for applications where it is necessary to precisely monitor and control rotary movement. They are especially useful in systems where position feedback is crucial, such as robotics, industrial automation, speed control systems, among others. The use of Hall effect encoders contributes to more precise and efficient control of the rotary movement, being a key part in many modern systems.

## DEVELOPMENT

### *USER INTERFACE*

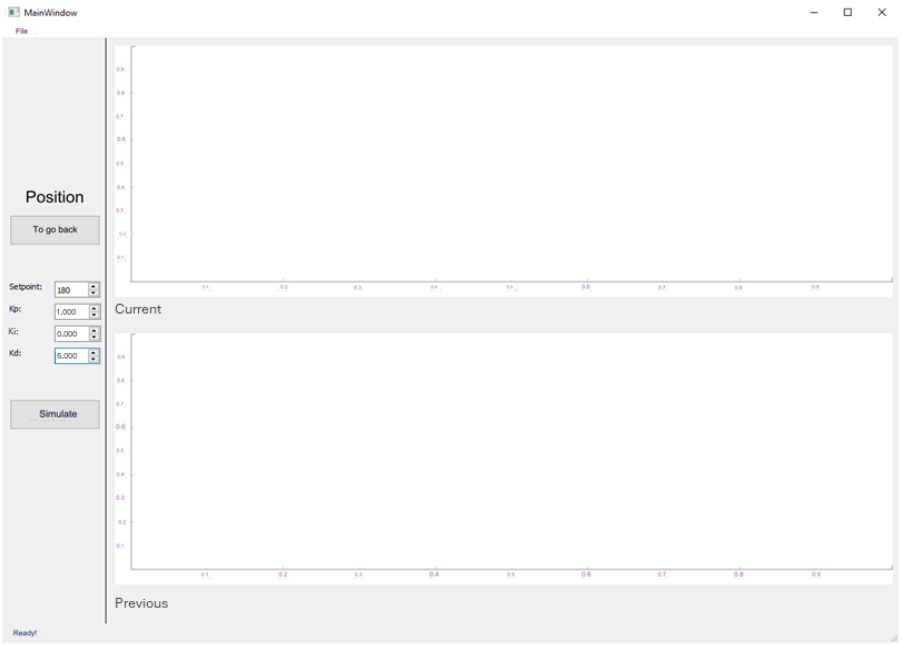
The process begins with user interaction through a graphical interface. The menu screen features three distinct buttons, each intended for a specific control: position control, speed control and comparison between phase advance, PD and PID controllers. The structure developed using object orientation in the Python language, and the PyQT5, with auxiliary from the book [6] allows easy navigation between pages. The "Menu" screen is displayed to the user as in the figure below.

Figure 4 — Interface menu.




There, the "Position" and "Speed" options open windows that allow you to configure and view the simulation. On the first two pages, for position and speed control, it is possible to enter parameters such as KP, KI and KD to configure the PID controller, in addition to the setpoint of the system. The simulation is started using a button that sends information to the microcontroller through serial communication, which will later be used to control the motor. The position and speed PID control screens are shown in the following figures:

Figure 5 — Interactive simulation environment for the position.




The fields to be completed refer to the proportional, integrative, and derivative parameters of a PID, and the Setpoint represents the control target value, as detailed below:

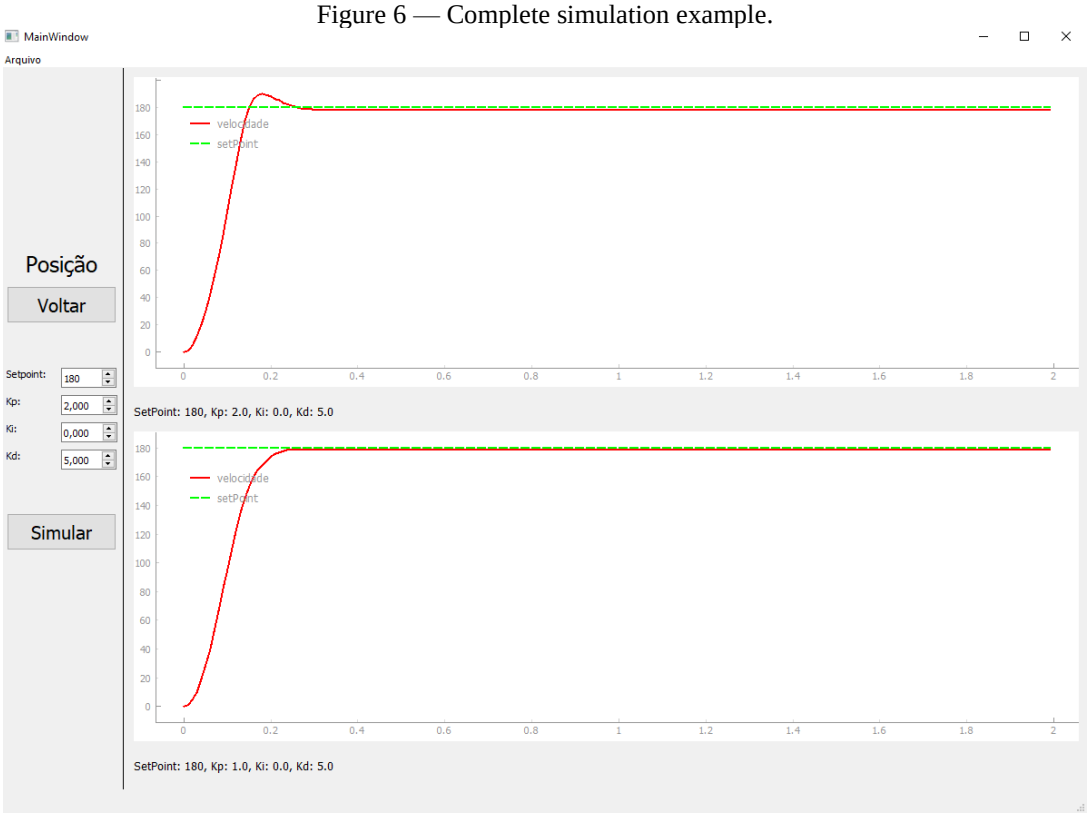
- KP (Proportional): The proportional parameter controls the system's proportional response to the difference between the system output and the reference value (setpoint). A larger KP value results in a stronger proportional correction, meaning the system output adjusts more quickly to the change in setpoint. However, a value that is too high can lead to instability or unwanted oscillations.
- KI (Integral): The integral parameter acts to correct errors accumulated over time. It is responsible to reduce steady-state error and improving system accuracy. A larger value of KI increases the influence of the integral and reduces the accumulated error but can also lead to slow responses and oscillations.
- KD (Derivative): The derivative parameter controls the response of the system to the rate of change of the controlled variable. It works to prevent large oscillations and smooth the system response, providing stability. A higher KD helps prevent spikes and dampen oscillations, but an excessively high value can introduce delays in the system's response.
- Setpoint: The Setpoint represents the desired value or reference for the controlled variable. In the context of a PID control, the objective is to adjust the system output to achieve and maintain the setpoint. The control loop continuously compares the actual system output to the Setpoint and adjusts the PID parameters to minimize error.

Inserting the parameters, it is possible to simulate the system. Using the “Simulate” button, a graphical response is generated of the behavior of the motor/flywheel assembly in relation to the parameters entered. This can be done from both the “Position” screen and the “Speed” screen.

Finally, the “Controllers” screen allows comparison between PD, PID and phase advance controllers, the first being calculated with the help of MatLab software and the last two calculated manually and detailed throughout the project. This screen has only one input data, setpoint. Once completed, simply click on one of the controller buttons for the motor to be evaluated and the response to appear graphically to the user on the screen.

An example of full test comparing the previous and current simulation is demonstrated below:

Figure 6 — Complete simulation example.

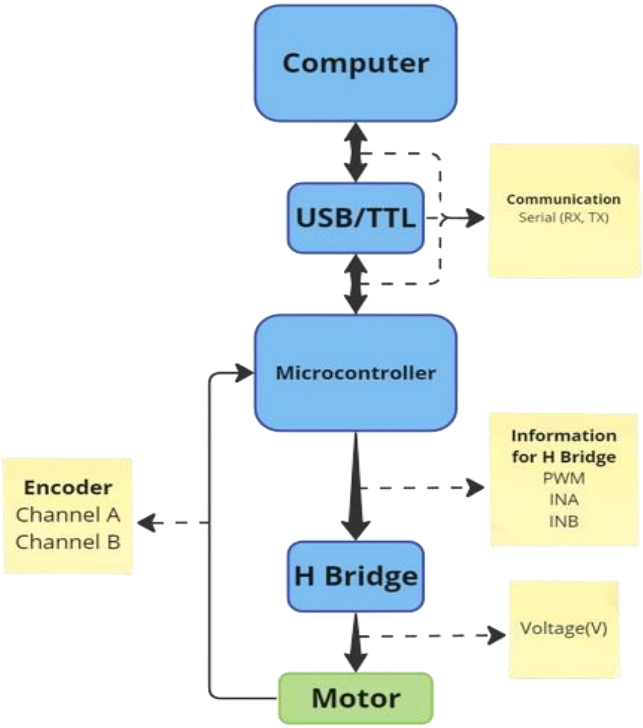



So, it is possible to save a specific test to reopen it in the future. This function is important for making comparisons between tests conducted at various times and recording them for future analyses.

## PROJECT STRUCTURE AND COMMUNICATION

In user-program interaction, the system can be configured to control the position and speed of a motor. The project structure diagram is presented in the following figure to facilitate the visualization of each element within the project and how they are interconnected:

Figure 7 — Project block diagram.




Communication between the computer, where the interface is located, and the microcontroller occurs via a USB TTL cable, using the UART communication protocol.

UART, Universal Asynchronous Receiver/Transmitter, is a method of communication between electronic devices that involves the transmission and reception of data in series, one bit at a time, through two wires: one for transmitting (TX) and the other for receiving (RX).

This is asynchronous communication, which means that there is no clock signal shared between the devices. Instead, devices must agree on a baud rate, called Baud Rate, which determines the speed of transmission. Before and after each data packet a start and stop are added to indicate the beginning and end of the transmission.

The microcontroller, upon receiving the information package, reads this data, processing it through the code developed in MPLAB, which contains the system control software.

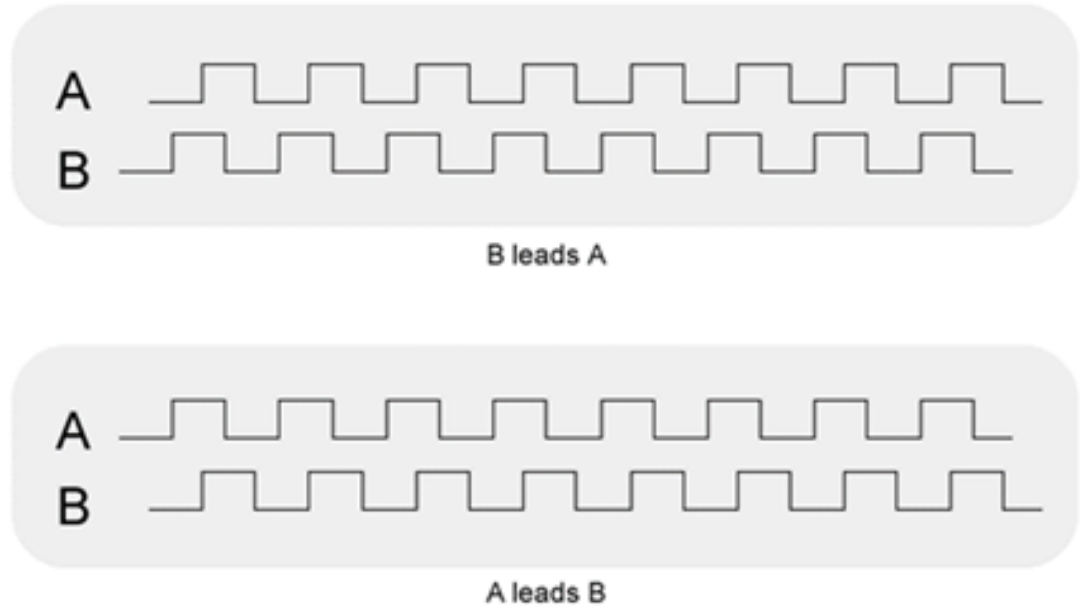
After treatment, a PWM (Pulse Width Modulation) signal is transmitted to an H-bridge. Compared to analog control methods, PWM is energy efficient as the device is on or off, minimizing power losses.

The H bridge, with “INA” and “INB” inputs, controls the direction of the motor, determining the clockwise or counterclockwise direction of movement. The resulting signal is converted into voltage (V) and transmitted to the motor to conduct the desired movement, whether for position or speed control, according to the settings previously entered by the user.

To close the control loop, an encoder monitors the motor through two channels, A and B, providing information about the direction of rotation. This data is sent back to

the microcontroller, allowing effective closed-loop control. The reading and interpretation of signals transmitted by channels A and B can be better understood by the image below. By combining the rising edge times of the two channels, it is possible to infer whether the motor rotates clockwise or counterclockwise:

Figure 8 — Quadrature of an encoder




Thus, encoder feedback is integrated into the control process. This allows the PIC to send commands to the engine and receive constant information about its status. Based on this feedback, the PIC dynamically adjusts the PWM signals sent to the H-bridge, optimizing efficiency, and ensuring real-time corrections to maintain the desired motor position.

With the system in closed loop, its response can be relayed to the computer, so that the user can graphically visualize the system's response to the previously entered parameters. It is noteworthy, therefore, that communication between the microcontroller and the computer is bidirectional. The information processed by the microcontroller is returned to the computer, completing the communication cycle.

### *MICROCONTROLLER PROGRAMMING LOGIC*

Before starting to develop the microcontroller programming, it will be necessary to determine which model will be used, this depends entirely on the requirements necessary for the project. Some of these requirements were mentioned previously, the microcontroller peripherals, these are the digital outputs and the PWM output, the UART communication and interruptions, both external and by timer present in the microcontroller. All these peripherals are in the PIC studied in Microcontrollers classes, the PIC16F1619, being a very versatile microcontroller as it is also compatible with the MCC (MPLAB® Code Configurator ) of the microchip, however one of the project requirements was not covered by this model, its SRAM memory was smaller than necessary to store the simulation information, therefore for the project it has to be a PIC

with the same peripherals, compatible with the MCC and has a memory greater than 2KB, this being the PIC16F18877, thus meeting all the project requirements.

With the microcontroller model determined, the next step is to begin developing the program. For this, MPLAB and the microchip's own MCC were used. The first step is to configure the microcontroller in the MCC and then the peripherals in the same environment.

Clock was configured, choosing the internal 32 MHz oscillator without any divider and without watchdog.

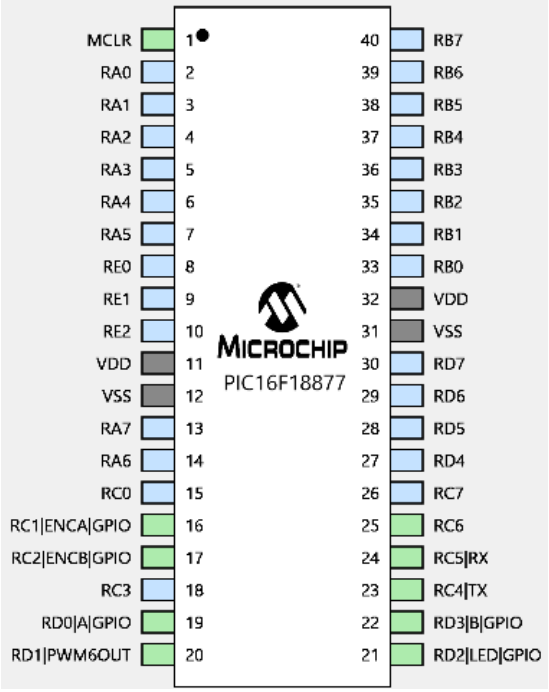
The first peripheral to be configured is the UART, called EUSART (Enhanced/ Addressable Universal Asynchronous Receiver Transceiver), just add it and configure it. A Baud Rate of 9600 was chosen due to the low error and the use of STDIO commands was enabled.

The next peripheral is the PWM output, adding which it is necessary to configure a timer together, defining its period to correspond to the frequency of the PWM signal. This period was chosen because the values used in the timer configuration correspond to an 8-bit value (0-255). And the frequency is lower than the maximum frequency of the motor driver.

And the last peripheral is the timer that will execute the interruption when it overflows. In the settings it is possible to determine the overflow period, this being the value of the controllers' sampling time of 10 ms.

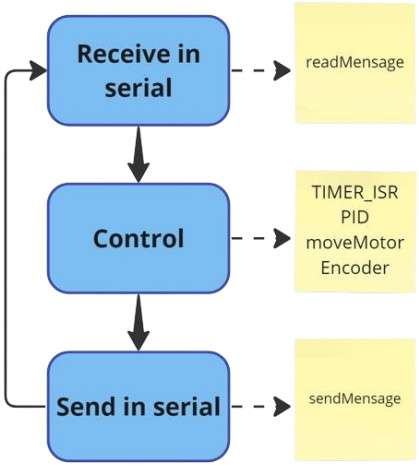
And the last part within the MCC is choosing and configuring the pins used, these are those of the peripherals such as the UART and PWM, the digital outputs for the H bridge (outputs) and the input pins (Input) for the encoder, which must be configured to have interrupt on the rising edge, IOC (Interrupt On Changing).

Figure 9 — Datasheet PIC16F18877



At this moment, the microcontroller has the capacity to do what was planned, however, to facilitate the development of the program, some functions were developed following the logic diagram:

Figure 10 — Microcontroller block diagram I.



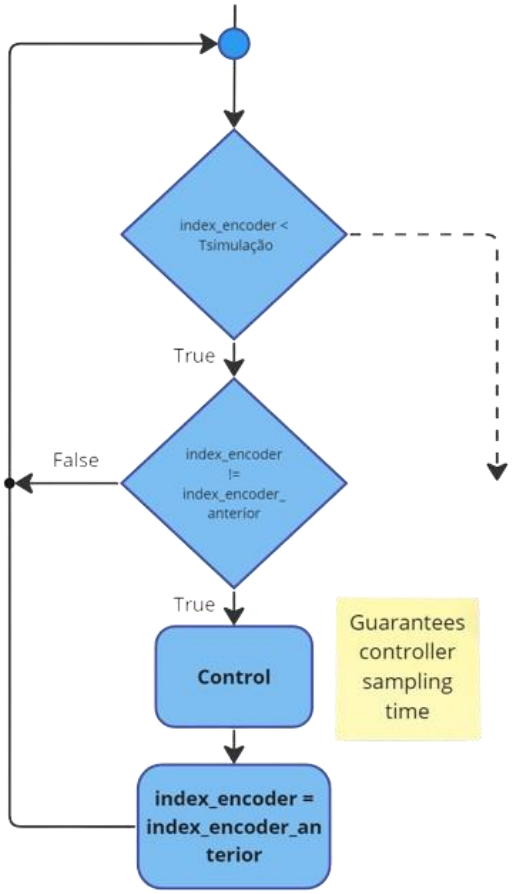

The first block is responsible for receiving the information coming from the simulator in serial form, the message sent has a standard format containing the mode, setpoint and, if necessary, the gains separated by commas, so the readMessage function has to be able to read the entire message, separating information and assigning it.

With the assigned data it is possible to move forward, the next step is to control the motor, for this it is necessary to develop 3 function, the first is a function that will receive an integer value, negative or positive, which will control the direction and PWM signal of the motor, in addition to controlling the motor dead zone.

Another function involves two ISR (interrupt service routine) of the encoder's external interrupt pins, these functions were designed in the following diagram to determine the amount and direction of rotation, adding, and subtracting a value from the variable.

Finally, it is necessary to develop another ISR, but now to interrupt the timer overflow, this function must be very brief and therefore cannot contain all the controllers' calculations. The following logic was developed, in which the interruption "locks" another function of the loop, which can only occur every 10ms, the sample time.

Figure 11 — Microcontroller block diagram II.




At this moment the basis for being able to control the plant is made, however the controllers have not been implemented, these will be implemented in the PID function, depending on the mode read in the serial, in readMessage , it will determine which controller will calculate the output from the error, using the value of the encoder and setpoint, applying it to the motor using moveMotor and at the same time saving the current state of the motor and updating the timer's ISR locking logic.

At this moment, the motor is being controlled, now, as the last step, just send all the data to the serial to display on the interface, for this we used sendMessage, which reads all the data stored during the control and the time of each moment.

In this the microcontroller programming was implemented successfully.

## *ENGINE MODELING AND SIMULATION*

### SYSTEM IDENTIFICATION

The first step in designing a controller is knowing what it should control. In mathematical terms, we must know the transfer function that will be worked on. In the case of the project, the transfer function of the motor system and the flywheel must be found.

As it is a DC motor, its modeling is well developed, and it is possible to find complete diagrams, such as the one shown in the following figure:

Figure 12 — Diagram of a plant with a DC motor.

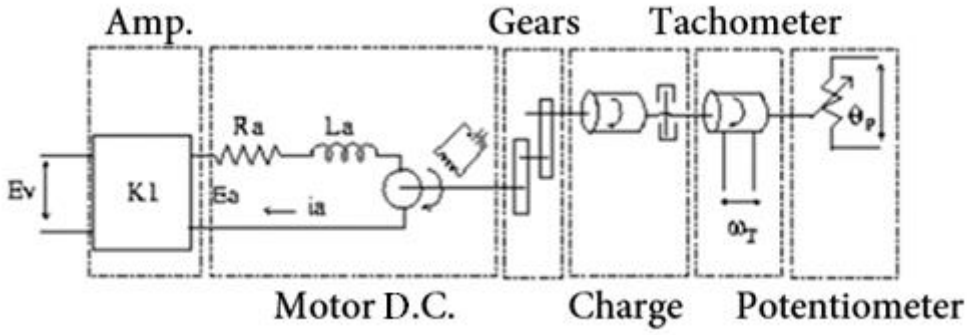
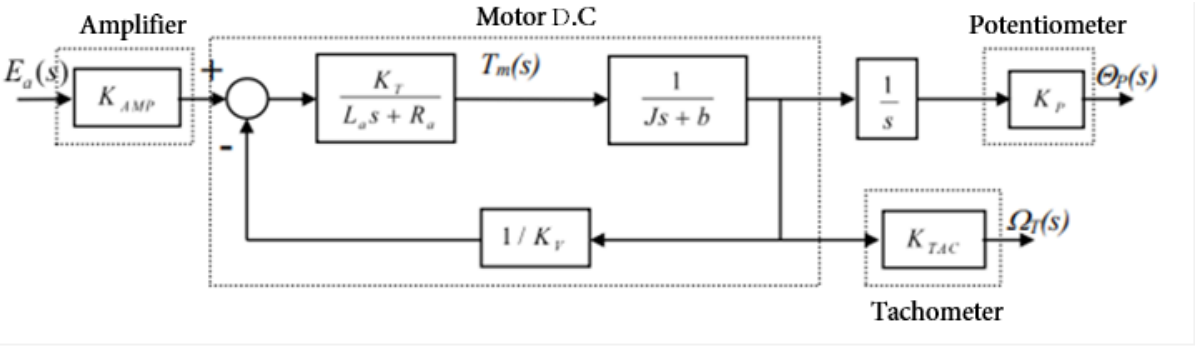



Figure 13 — Block diagram of the plant with the DC motor.




By studying the model and its simplifications, it is possible to approximate the engine model for speed control in a first-order system, and for position control in a second-order system, described below:

$$G_\omega(s) = \frac{\Omega_T(s)}{E_V(s)} = \frac{K_w}{Ts + 1}$$

$$G_\theta(s) = \frac{\theta_P(s)}{E_V(s)} = \frac{K_\theta}{s(Ts + 1)}$$

There are several ways to acquire this transfer function. One of them is through modeling the motor, using its physical and electrical characteristics to find the desired function. However, the engine in question does not have the necessary information for this modeling, making this method unfeasible for the application in question. Therefore, it was decided to obtain the transfer function through the test method, that is, surveying the engine curve. For this test, it was decided that 5 steps will be conducted with the following values:

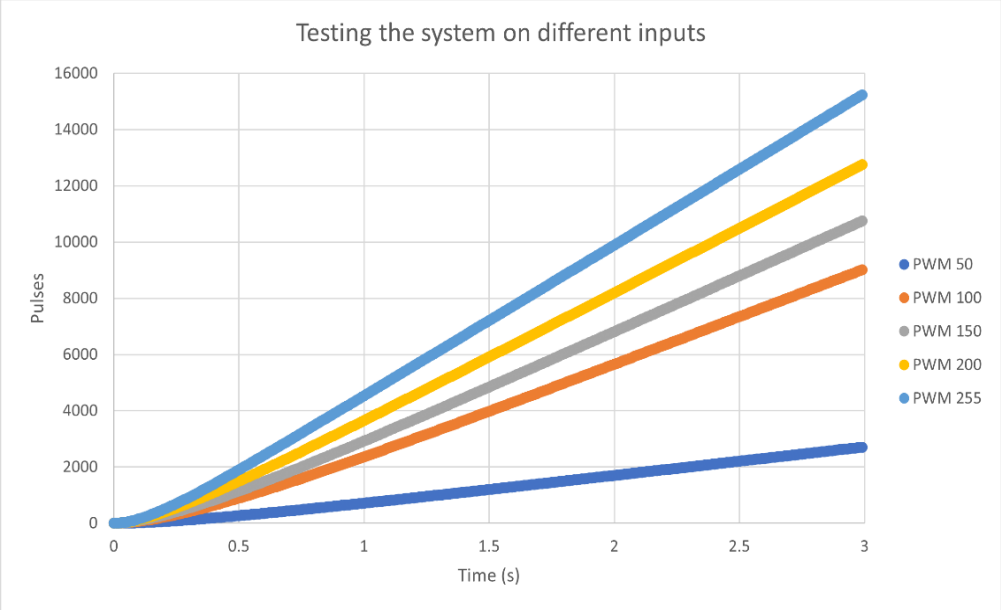
Table 1 — Test input data.

| PWM (8-bit) | Percentage (%) |
|---|---|
| 50 | 19.6% |
| 100 | 39.2% |
| 150 | 58.8% |
| 200 | 78.4% |
| 255 | 100% |

With this data, it is possible to define the test input (U).

The values were selected based on the microcontroller's PWM resolution. Once the inputs are defined, the experiment can be conducted. The proposal consisted of recording the number of encoder pulses over 3 seconds, a period sufficient to stabilize the system. Data analysis will be approached in four ways: two analytical analyses, using position and velocity data, and two computational techniques, using MatLab software with the same data sets. At the end of the tests, the following data were recorded:

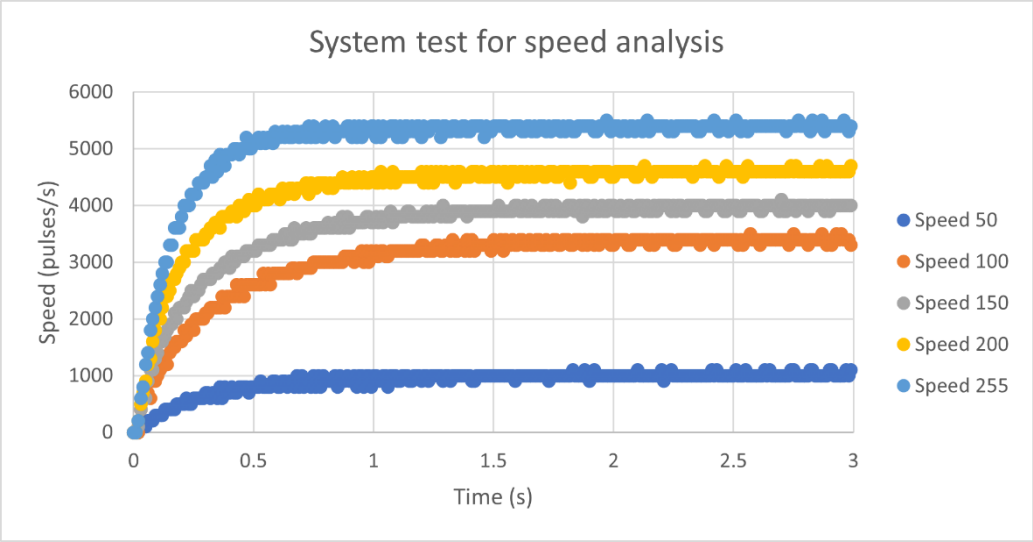
Figure 14 — Graph of position tests.




With the recorded data, the units of the quantities were defined to reduce the processing needed in the microcontroller. Thus, the speed unit will be pulses per second (pulses/s) and position in pulses (pulses).

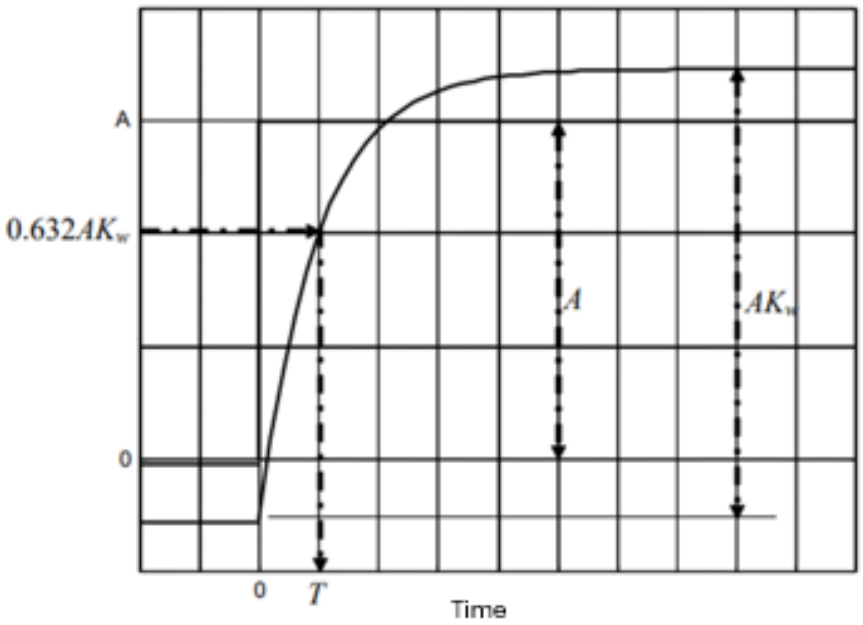
Starting with the speed analysis, the data was processed, arriving at the following graph:

Figure 15 — Graph of speed tests.




With this data, a separate analysis of each output was conducted, following the model:

Figure 16 — Calculation model for speed tests.




Using the data and concepts in the example above, we obtain the following transfer functions and their average model:

Table 2 — Velocity transfer functions by the analytical method

| Transfer Function | Time constant(s) | Gain |
|---|---|---|
| $G_{50} = \frac{20}{0.3s + 1}$ | 0.3 | 20 |
| $G_{100} = \frac{34}{0.3s + 1}$ | 0.3 | 34 |
| $G_{150} = \frac{26.67}{0.27s + 1}$ | 0.27 | 26.67 |
| $G_{200} = \frac{23}{0.19s + 1}$ | 0.19 | 23 |
| $G_{255} = \frac{21.176}{0.18s + 1}$ | 0.18 | 21.176 |
| $G_M = \frac{24.97}{0.248s + 1}$ | 0.248 | 24.97 |

Using another method of obtaining, using the MatLab software and its system identification part, we arrive at the following functions:

Table 3 — Software speed transfer functions

| Transfer Function | Time constant(s) | Gain |
|---|---|---|
| $G_{50} = \frac{20.128}{0.321s + 1}$ | 0.321 | 20.128 |
| $G_{100} = \frac{33.697}{0.353s + 1}$ | 0.353 | 33.697 |
| $G_{150} = \frac{26.26}{0.295s + 1}$ | 0.295 | 26.26 |
| $G_{200} = \frac{22.776}{0.21s + 1}$ | 0.21 | 22.776 |
| $G_{255} = \frac{21.04}{0.16s + 1}$ | 0.16 | 21.04 |
| $G_M = \frac{24.7802}{0.2678s + 1}$ | 0.2678 | 24.7802 |

When analyzing the results obtained, the average transfer functions of the two methods exhibit similar parameters, with a percentage error of 0.76% for the gain and 7.4% for the time constant. This way, it becomes possible to consider an average model through velocity analysis.

$$G(s) = \frac{24.8751}{0.2579s + 1}$$

In addition to analyzing the speed data to estimate the transfer function, this project proposes an additional approach to obtain the characteristic curve. The engine

position data will be used, allowing subsequent comparison of the results obtained by both methodologies. The process will follow a similar approach to that conducted for speed, followed by the analysis in MatLab, according to the model presented below:

Figure 17 — Calculation model for position tests

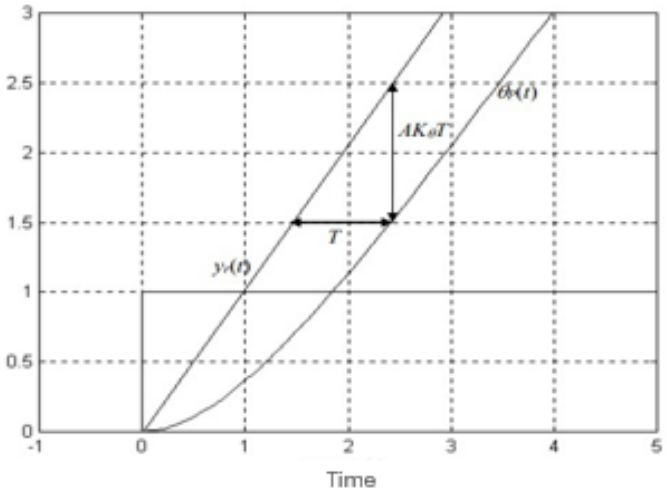



Table 4 — Position transfer functions by analytical method and software.

| Analytics | | | *Software* |
|---|---|---|---|
| Transfer Function | Cte. Of time | Gain | Transfer Function |
| $G_{50} = \frac{19.93}{s(0.3s+1)}$ | 0.3 | 19.93 | $G_{50} = \frac{63.67}{s^2+3.342s+4.64*10^{-9}}$ |
| $G_{100} = \frac{33.68}{s(0.31s+1)}$ | 0.31 | 33.68 | $G_{100} = \frac{88.53}{s^2+2.748s+1.368*10^{-9}}$ |
| $G_{150} = \frac{26.17}{s(0.27s+1)}$ | 0.27 | 26.17 | $G_{150} = \frac{74.97}{s^2+2.978s+2.315*10^{-7}}$ |
| $G_{200} = \frac{23.4}{s(0.2s+1)}$ | 0.2 | 23.4 | $G_{200} = \frac{97.9}{s^2+4.43s+2.003*10^{-9}}$ |
| $G_{255} = \frac{20.78}{s(0.16s+1)}$ | 0.16 | 20.78 | $G_{255} = \frac{135.7}{s^2+6.521s+4.108*10^{-10}}$ |
| $G_M = \frac{24.794}{s(0.248s+1)}$ | 0.248 | 24.794 | $G_M = \frac{92.154}{s^2+4.002s+4.8*10^{-8}}$ |

The first important conclusion observed from the comparison of the methods is the transfer function without the free integrator obtained using MatLab tools, different from the simplified model obtained analytically. The model without the free integrator will be used for the analytical calculation of the PID controller, which will be discussed in the future.

The second possible conclusion is: considering that the integrated velocity transfer function results in the position transfer function for the same input, we arrive with similar transfer functions, with a time constant error of 3.8% and gain of 0.33%.

With all the data obtained and calculated, it is possible to obtain an average model of the engine studied:

$$G(s) = \frac{24.8345}{s(0.2579s + 1)}$$

## SYSTEM VALIDATION

By obtaining the transfer function that describes the system mathematically, it is possible to make a comparison between the simulated response and the real response of the system. Below are some tests that include the curves corresponding to simulations and practical execution.

Figure 18 — Comparison of position and real models.

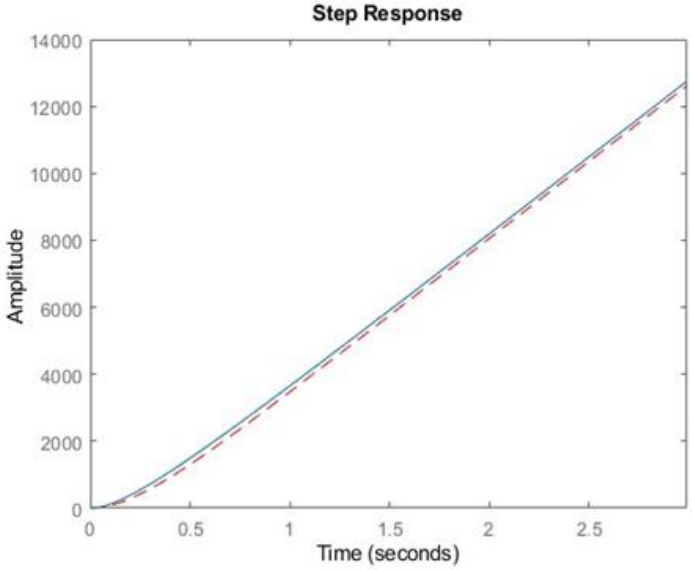
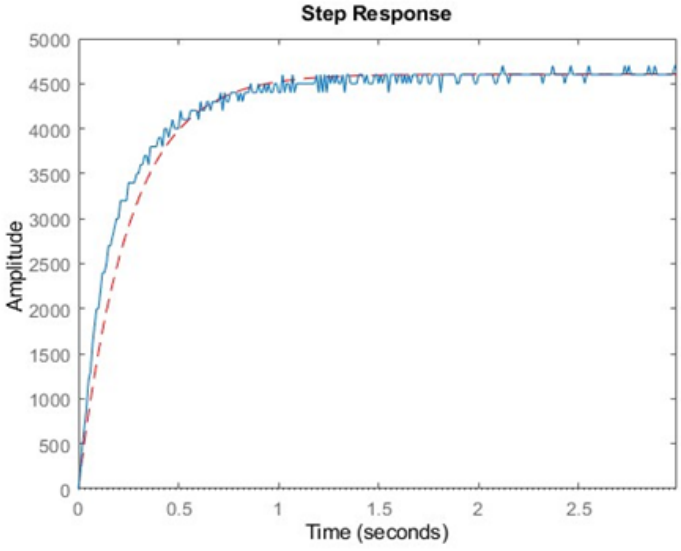



Figure 19 — Comparison of speed and real models.



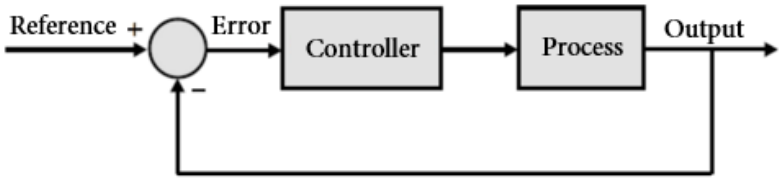
The blue curve represents what the engine performed, and in dashed red is the simulation using the transfer function obtained in the tests.

## SYSTEM CONTROL PROPOSAL

The primary purpose is to compare the calculated controllers and those designed with the help of MatLab. These include a phase advance controller, a PID controller and a PD controller, the latter being designed in MatLab.

Firstly, all controllers must follow the same block diagram:

Figure 20 — Block diagram




The first controller designed will be the phase advance for controlling the motor position. For this, the project requirements were determined: a settling time ( $T_S(2\%)$) of 1 second and an overshoot ($M_{pt}(\%)$ ) of 1%. Knowing this, it is possible to start designing the controller. Knowing that:

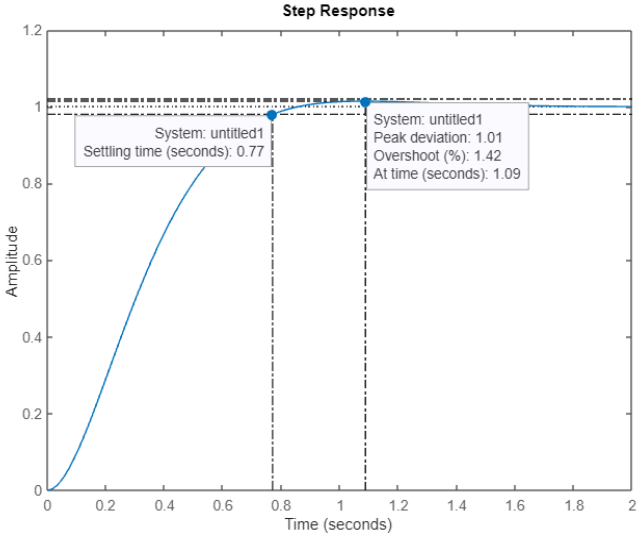
$$G_c(s) = \frac{k_c(s + z_c)}{s + p_c}, z_c < p_c$$

The following controller was obtained from the parameters obtained by the tests and requirements:

$$G_c(s) = \frac{0{,}2632(s + 4)}{s + 8{,}46}$$

To evaluate whether the calculated controller meets the design parameters, it is possible to simulate its response in MATLAB.

Figure 21 — Controlled system response



Another proposed controller is the PID, but for this it will be necessary to use another model, one that contains two poles other than 0, as mentioned previously:

$$G_M = \frac{92{,}154}{S^2 + 4{,}002s + 4{,}8 * 10^{-8}} = \frac{92{,}154}{(s + 4{,}002)(s + 1{,}2 * 10^{-8})}$$

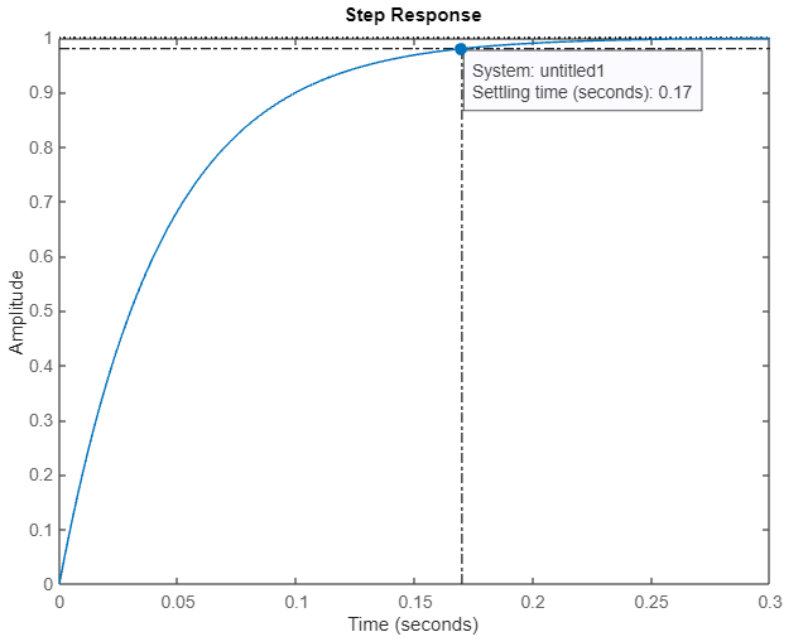
The PID controller can be written as seen below, and at the end of the process the values of Kp, Ti and Td must be defined.

$$G_{PID} = k_p * \left(1 + \frac{1}{T_i s} + T_D s\right) = k_p \left(\frac{T_i T_D s^2 + T_i s + 1}{T_i\, s}\right)$$

The first step is to find the values of Ti and Td to "cancel" the plant poles. Admitting the designed controller, we have the following controller and the response simulated in MatLab:

$$G_{PID} = (1 + \frac{1}{83375000 * s} + 0{,}2499 * s)$$

Figure 22 — Controlled system response

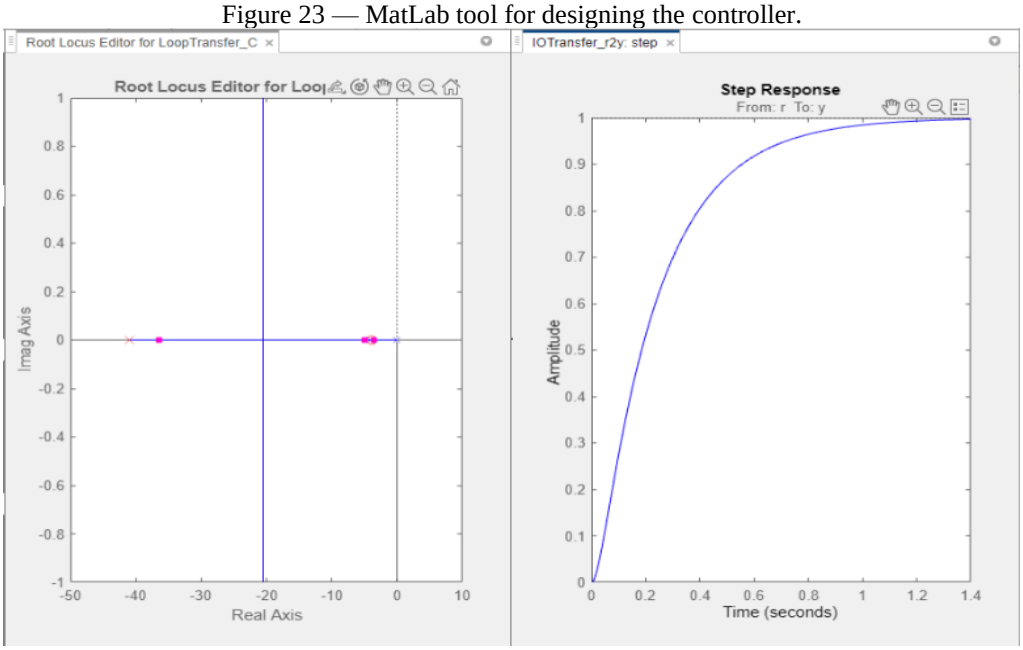



The result obtained is satisfactory for the controller design. In addition to the phase advance and PID controllers mentioned, a comparison will be presented with a PD controller developed using MatLab tools, specifically RLTOOL.

The optimization of the result was conducted according to the project requirements, considering the sensitivity to the controller's gain. The results obtained include the specific configuration of the controller and the corresponding output of the plant controlled by that device.

$$G_C(s) = 0{,}17016 * \frac{(1 + 0{,}25s)}{(1 + 0{,}024s)}$$

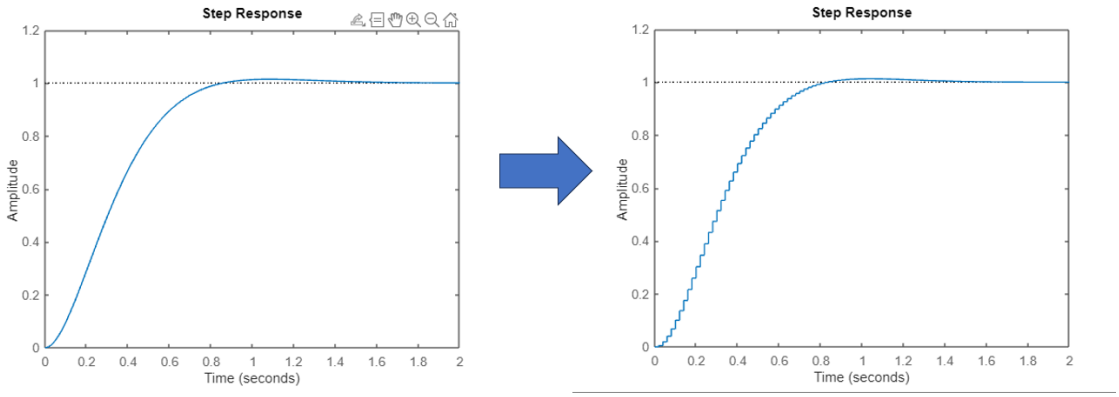
Figure 23 — MatLab tool for designing the controller.




## EMBEDDED CONTROL

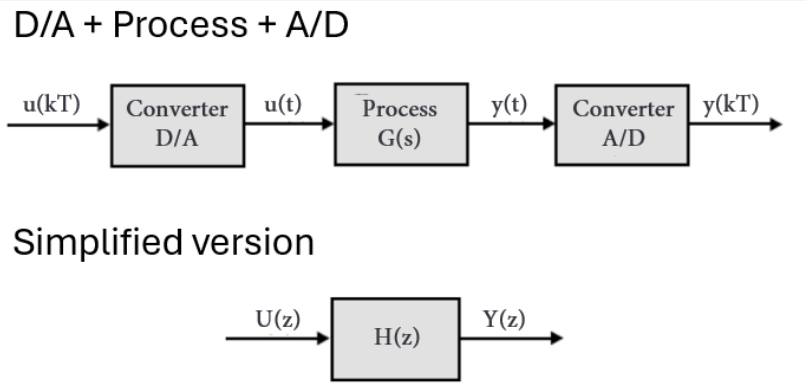
To integrate a controller into a microcontroller or PLC, for example, it is necessary to discretize the designed controller in continuous time. This discretization implies that the controller will be updated at specific intervals defined by T, called sampling time. Below is an example of a discretized system:

Figure 24 — Representation of the discrete system




To do this, the transfer function of our controller will be used, and the discretization will be done using the Z transform and the simplification of the diagrams with the analog/digital and digital/analog converters, as in the image and equation below:

Figure 25 — Block diagram of a discretized system.




With $H(z) = (1 - z^{-1})\,\mathcal{Z}\left[\frac{G(s)}{s}\right]$

Using this method and MATLAB's "2d" tool we arrive at the following discretized controllers:

$$H_{af}(z) = \frac{0{,}2632 * z - 0{,}2531}{z - 0{,}9189}$$

$$H_{PD}(z) = \frac{1{,}772 * z - 1{,}715}{z - 0{,}6592}$$

$$H_{PID}(z) = \frac{1{,}509 * z^2 - 2{,}94 * z + 1{,}431}{z^2 - 1{,}607 * z + 0{,}6065}$$

The next step is to find the difference equation using algebraic manipulations and the delay property. $\mathcal{Z}\{u[k-1]T\} = z^{-1}U(z)$. Using the mentioned property, it is possible to obtain the following difference equations:

$$y_{af}(kT) = 0{,}2632\,u[kT] - 0{,}2531u\,[(k-1)T] + 0{,}9189y_{af}[(k-1)T]$$

$$y_{PD}(kT) = 1{,}772u\,[kT] - 1{,}715\,u[(k-1)T] + 0{,}6592y_{PD}[(k-1)T]$$

$$y_{PID}(kT) = 1{,}509u[kT] - 2{,}94u[(k-1)T] + 1{,}431u[(k-2)T] + 1{,}607y_{af}[(k-1)T] - 0{,}6065y_{PID}[(k-2)]$$

With "u" being the input, the error, and "y" being the system output, the PWM, applied to the motor.

With these difference equations it is possible to implement them on the microcontroller and evaluate them to compare them with the simulated results, as demonstrated in the following graphs:

Figure 26 — System comparison in Phase Advance

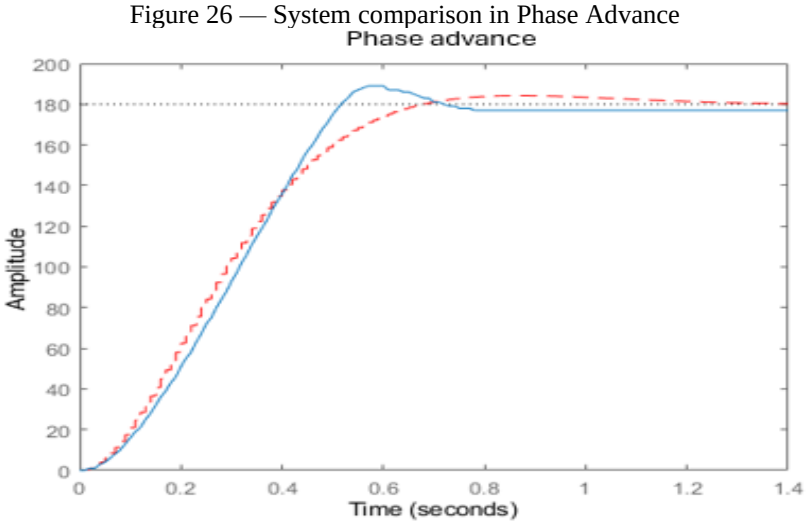



Figure 27 — Comparison of the system in Proportional and Derivative

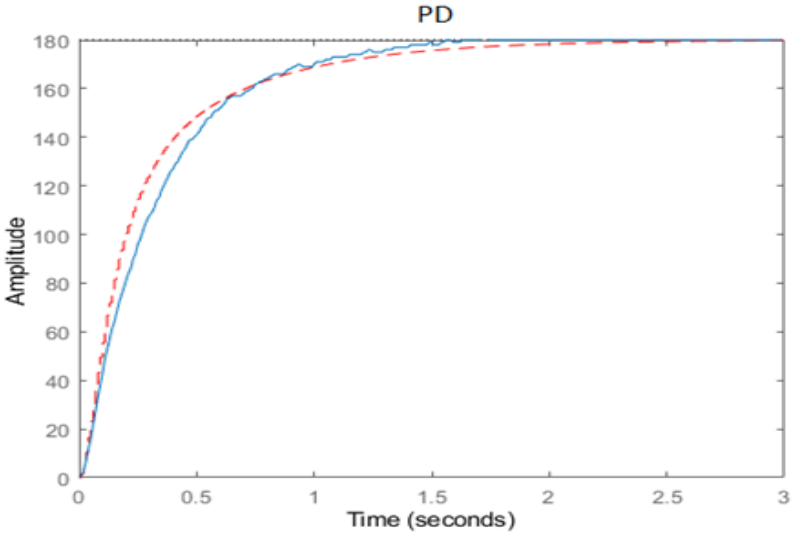
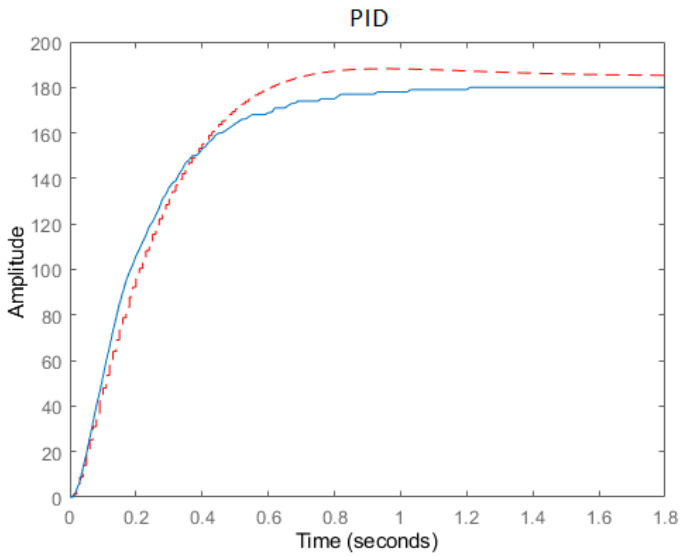



Figure 28 — System comparison in PID



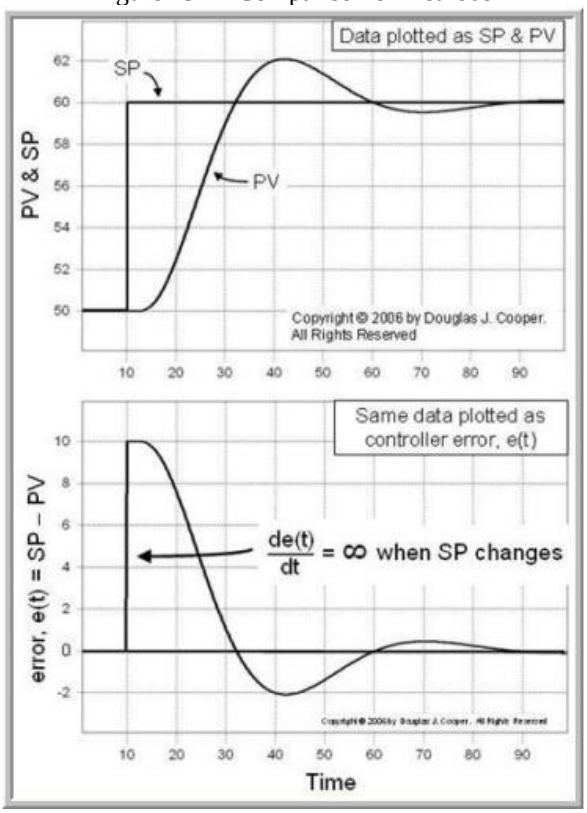
The dashed red line represents the simulated curve, and the blue line represents the real system curve.

Another discretized control that is important to mention is included in the first part of the project, in which the gains of a PID controller are interactive. In this part, there is a controller implemented as follows:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t)$$

The integrative component is obtained by the sum of the product of the error and the time variation (Δt), while the derivative component results from the difference divided by the same time interval. Typically, this difference is calculated between the current and previous error, but this approach can lead to a derivative that tends to infinity. To overcome this problem, we chose to vary the process variable, using difference between the position instead of the difference between the errors. Thus, the subtraction is conducted between the current position and the previous position, inverting the behavior of the derivative. To correct this inversion, the derivative portion is subtracted instead of added, as evidenced in the following comparison:

Figure 29 — Comparison of methods




This way, the following function is obtained:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{d}{dt} PV(t)$$

## RESULTS AND DISCUSSION

When looking at the finished project, the application of both the analytical method and the software method for speed or position analysis resulted in obtaining engine models that are close, presenting small deviations, as previously mentioned. Therefore, both approaches demonstrated effectiveness in meeting the proposed objective, which was to obtain the engine model through the tests conducted.

A crucial point identified in the analyzes is the presence of disparities in the system's response, influenced by a variety of factors, among which the slack in the reduction box stands out. This slack has a significant impact on the dynamics of the system. Considering the impossibility of eliminating this effect, it is suggested the evaluation and adoption of an alternative reduction box, aiming to improve the agreement of the results obtained with the established expectations.

Another relevant aspect addressed refers to the motor's dead zone, characterized by a voltage range between 0V and a specific value. During this interval, the engine remains inert due to mechanical considerations, associated with static friction and moments. This identification provides valuable insights for understanding and improving the system's behavior in specific situations.

Despite the subtle discrepancies observed, the project fully met expectations, allowing efficient comparison of efficiency between different controllers and methods. Furthermore, it provided a comprehensive understanding of the concepts covered in the relevant disciplines, thus consolidating the success of the research in achieving its objectives and contributing to the advancement of knowledge in the area in question.

## CONCLUSIONS

The project covered in this report represents a significant advance in the practical and interdisciplinary application of the concepts learned in the disciplines of Instrumentation, Microcontrollers, Control Systems I, Object Oriented Programming and Database. The creation of an interactive simulator to control the speed and position of a motor coupled to an inertia wheel is a milestone that highlights the convergence of hardware and software in engineering, providing a practical and holistic experience for students.

The primary objectives of the project were successfully achieved, highlighting the creation of a virtual environment that enables the practical application of the theoretical concepts studied. The use of the PIC16F18877 to control the engine, together with the implementation and validation of the speed and position control systems, demonstrates the effectiveness of the approach adopted. The project phases, from the development of the virtual environment to the construction of the hardware, were structured in a logical and coherent manner.

The developed simulator provides an effective platform for experimentation and understanding key concepts such as PID control, mathematical system identification and

microcontroller programming. The ability to simulate different closed control loops in the same device is a valuable contribution, allowing direct comparison between different control strategies. This comparative analysis is crucial to optimizing desired control performance and deepening understanding of control principles.

The development of this interactive simulator represents not only the practical application of acquired knowledge, but also highlights the importance of interdisciplinary integration in the training of engineering professionals. The approach adopted not only provides a deeper understanding of theoretical concepts, but also prepares students to face real-world challenges, where collaboration between different areas of knowledge is essential.

## REFERENCES


[1] A. Ribas Neto, M. Fiorin and T. Dequigiovani, "Projeto Integrador na Engenharia de Controle e Automação", in III Seminário Integrado de Ensino, Pesquisa e Extensão do IFC, Santa Catarina, Brazil, 2016-06-24. Santa Catarina: Instituto Federal Catarinense, 2016, pp. 1–8. [Online]. Available: https://eventos.ifc.edu.br/seminariointegrado/wpcontent/uploads/sites/4/2017/06/Projeto-Integrador-Na-Engenharia-De-Controle-EAutoma%c3%a7%c3%a3o.pdf [Accessed 2022-04-12].

[2] A. Said, ‘Comparison between FLC and PID Controller for speed control of DC Motor’, 03 2022.

[3] N. A. Kheir et al., “Control systems engineering education,” Automatica, vol. 32, no. 2, pp. 147–166, Feb. 1996, doi: https://doi.org/10.1016/0005-1098(96)85546-4.

[4] R. Heradio, L. de la Torre, and S. Dormido, “Virtual and remote labs in control education: A survey,” Annual Reviews in Control, vol. 42, pp. 1–10, 2016, doi: https://doi.org/10.1016/j.arcontrol.2016.08.001.

[5] S. Balamurugan and A. Umarani, “Study of Discrete PID Controller for DC Motor Speed Control Using MATLAB,” 2020 International Conference on Computing and Information Technology (ICCIT-1441), Sep. 2020, doi: https://doi.org/10.1109/iccit-144147971.2020.9213780.

[6] HARWANI, BM Qt5 Python GUI Programming Cookbook Building responsive and powerful cross- platform applications with PyQt. [ sl.] Birmingham; Mumbai Packt July 2018.

[7] LIN, Paul-l-Hai; HWANG, Sentai; CHOU, J. **Comparison on fuzzy logic and PID controls for a DC motor position controller.** In: Proceedings of 1994 IEEE Industry Applications Society Annual Meeting, Denver, CO, USA, 1994. p. 1930-1935 vol.3. DOI: 10.1109/IAS.1994.377695.

[8] MAHMUD, M.; MOTAKABBER, SMA; ALAM, AHMZ; NORDIN, AN **Adaptive PID Controller Using for Speed Control of the BLDC Motor.** In: 2020 IEEE International Conference on Semiconductor Electronics (ICSE), Kuala Lumpur, Malaysia, 2020. p. 168-171. DOI: 10.1109/ICSE49846.2020.9166883.