OPEN ACCESS

International
Journal
Of Modern Engineering Research (IJMER)

PID Controller Embedded in Industrial PLC Applied to Level Control in Tanks

Antonio Henrique dos Santos Ribeiro[1], Bruno França Coelho[1], Evandro Martins Araújo Filho[1], Ana Caroline Meireles Soares[1], José Pinheiro de Moura[2], Patrícia Helena Moraes Rêgo[2], João Viana Da Fonseca Neto[1]

**Department of Electrical Engineering / Federal University of Maranhão /Brazil*
*** Center of Technological Sciences/ State University of Maranhão, Brazil*

ABSTRACT: *The present work addresses a methodology of Proportional-Integral-Derivative (PID) controllers applied in control systems in a two-tank plant at different levels using a Programmable Logic Controller (PLC). The procedure for elaborating the control programming logic in the ladder language is presented, and the results obtained through simulation are used to evaluate the performance of the controller tuning that is implemented in the PID controller block embedded in the PLC in order to maintain the levels of tanks 1 and 2 at their maximum. Tank 1 is supplied by the reservoir and tank 2, in turn, is supplied by tank 1 by the gravity action (the tank 2 inlet is equal to the outlet). A PI controller was designed for the system control. The PI controller was embedded in the PLC of the tank plant, and the results were analyzed with the system control.*

I. INTRODUCTION

Historically, the use of classic controllers has solved most of the control problems in industrial processes. However, in classical theory, plants or processes are treated as linear variables, otherwise, they are linearized around a point of operation [6]. In real situations, these linearity assumptions may not be sufficiently comprehensive or accurate to allow the representation of the plant over its entire range of operation. Thus, the development of non-linear tools for the control of these processes is fundamental to guarantee the quality and operational safety in industry. As a possible solution to the control problem in nonlinear processes, one can use intelligent systems [7].

The proportional-integral-derivative (PID) controller combines the advantages of the PI and PD controller. The integral action is directly linked to the accuracy of the system being responsible for the null error in a permanent regime [14]. The destabilizing effect of the PI controller is counterbalanced by the derivative action that tends to increase the relative stability of the system, while making the system response faster because of its anticipatory effect [2].

In this paper the PID controller technique was adopted, since it is a problem that can be solved by differential equations, and because it is an available method in several industrial PLCs.The fundamental objective of the implemented control, is to maintain the control of the level of tanks 1 and 2 at their maximum, thus avoiding overflow and empty tanks, guaranteeing the performance both in the servo and the regulatory problem.

The implementation of a control methodology based on PID controllers on large scale PLCs, which improves the automation of industrial equipment, is the main contribution of the present work. The development of the design and the implementation of a PI controller in an industrial PLC for control systems are presented. A fluid level plant composed of two tanks and a reservoir is used to test and validate the proposed PI controller model. In addition, a comparison of the results obtained before and after the implementation of the PI controller in the PLC of the study plant is presented.

The paper is organized in sections as follows: Section II presents a description of the tank level system, its operation and the modeling of the system with mathematical formulations. The design of the control system and its characteristics are discussed in Section III. The practical and computational experiments, in addition to the results obtained, are shown in Section IV, and finally the conclusion is presented in Section V.

II. DESCRIPTION OF THE TANK LEVEL SYSTEM

The tank system consists of: a pump; two continuous level sensors; switches for activation of the system; one Compact Logix L32E PLC; a computer; and four contactors.
The dimensions and characteristics of the tanks are presented in Table 1.

Table 1: Characteristics of the tanks

DIMENSIONS	RESERVOIR	TANK 1	TANK 2
Height (cm)	12	12	12
Base (cm)	24×12	12×12	12×12
Capacitance (cm^2)	288	144	144

Figure 1 shows a graphical illustration of the small-scale tank system used in the simulation and validation of the adopted methodology.

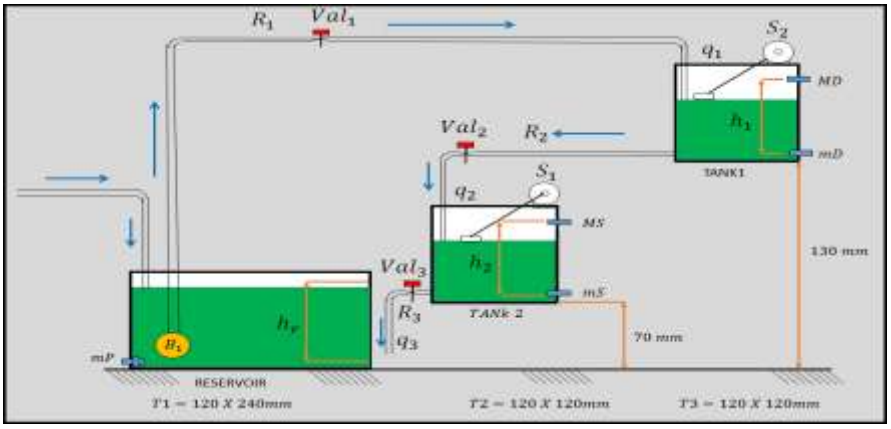

Figure 1: Graphic illustration of the plant

2.1 Plant Modelling

At the beginning of the plant process shown in Figure 1, tanks 1 and 2 are empty. The reservoir pump is driven to supply tank 1 until it reaches the maximum level, in the sequence tank 2 is started up, and it is supplied by tank 1 via gravity until it also reaches the maximum level (during the filling of tank 2, tank 1 is simultaneously supplied in the same tank supply flow 2), upon reaching the maximum level of tank 2, the outflow of tank 2 starts, this being equal to the inflow rate. The system works with the integrated process between the tanks, i.e., the reservoir is considered always full and tanks 1 and 2 are kept at the maximum levels, so the flow of the pump 1, after tank 1 and tank 2 are supplied, is the same as the outflow from tanks 1 and 2 [1].

The events occur in the following sequence: valve 1 is opened, pump 1 is connected, tank 1 is filled, tank 1 reaches maximum level, valve 2 is opened, tank 2 is filled, pump 1 flow is reduced (keeping equal the output of tank 1), filling tanks 1 and 2 simultaneously reaches maximum level of tank 2, and finally, valve 3 is opened.

2.2 System of Continuous Variables

Real plant processes are mostly complex, and it is difficult to develop models that represent well their operation. For the mathematical modeling of the Continuous Variable System (CVS), the resistance *R* to the liquid flow in the pipeline or restriction is taken into account, which is defined as the variation in the level difference required to cause the unitary variation in the flow rate given by [11]

$$R = \frac{dh}{dq}, \tag{1}$$

where *dh* the variation in the level difference and *dq* the variation in the flow in volume.
The capacitance C of a reservoir is defined as the change in the amount of stored liquid required to cause a potential unit change according to Equation (2) [11]

$$C = \frac{dv}{dh}, \tag{2}$$

where *dv* is the change in the amount of stored liquid.

The proposed plant was developed following an example found in [11] for a second order level system. Its modeling is carried out by linear differential equations obtained from the physical nature of the system, as shown in Figure 1. Three containers were used in translucent acrylic, one is the reservoir and the other two are where the levels are effectively controlled. The reservoir has been dimensioned according to the nominal flow of the actuator (engine) and it is compatible with the dimensions of the sensor. Each plant parameter is defined as follows:

The actual system parameters are: R2 = 0.5 s/cm^2; R3 = 0.4 s/cm^2; C2 = 144cm^3; e C3 = 144cm^3. The input was defined as q_1 and the outputs are at heights h_r and h_1

Balance conditions for tank 1

$$\frac{dh_1}{dt} = \frac{1}{C_3} q_1 - \frac{1}{C_3 R_2} h_2 . \tag{3}$$

Balance conditions for tank 2

$$\frac{dh_2}{dt} = \frac{1}{C_2 R_3} h_2 - \frac{1}{C_2 R_2} h_1 . \tag{4}$$

This system can be represented in state space by the following equations

$$\dot{x} = Ax + Bu , \tag{5}$$

$$y = Cx + Du . \tag{6}$$

Where *x* is the state vector, u is the input, and y is the output. Thus obtaining the State Space matrices.

$$\begin{bmatrix} \dot{h}_2 \\ \dot{h}_1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{C_2 R_3} & \frac{1}{C_2 R_2} \\ 0 & -\frac{1}{C_3 R_2} \end{bmatrix} \begin{bmatrix} h_2 \\ h_1 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{C_3} \end{bmatrix} q_1 , \tag{7}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} h_2 \\ h_1 \end{bmatrix} . \tag{8}$$

Substituting the parameter in Equation (7) one has

$$\begin{bmatrix} \dot{h}_2 \\ \dot{h}_1 \end{bmatrix} = \begin{bmatrix} -0.001736 & 0.01389 \\ 0 & -0.01389 \end{bmatrix} \begin{bmatrix} h_2 \\ h_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.006944 \end{bmatrix} q_1 . \tag{9}$$

2.3 Discrete -Time Linear System

Considering the small-scale tank system shown in Figure 1 and mathematically formulated in CVS according to Equation (7), this can be described by the second-order discrete-time linear system, according to Equation (10).

For the mathematical formulation of the linear system of discrete time, the sampling interval equal to 1s was adopted.

$$h_{k+1} = \begin{bmatrix} 0.9828 & 0.01367 \\ 0 & 0.9862 \end{bmatrix} h_k + \begin{bmatrix} 4.773 \times 10^{-5} \\ 0.006896 \end{bmatrix} q_1 . \tag{10}$$

The system involves two states $h_k = \begin{bmatrix} h_2(k) & h_1(k) \end{bmatrix}^T$ corresponding to the water filling levels of the tanks 1 and 2 with the initial condition $h(0) = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$. The control input is the filling flow of tanks 1 and 2. The sensors mS, MS, mD and MD are the measurements of the first and second states.

2.4 Discrete-Event Systems

According to some researchers, the year 1804 is considered, when Jacquard invented the loom machine with perforated cards, the "beginning" of the control of sequential systems that are a class of DES (Discrete Event System). The Boolean algebra, which is one of the mathematical bases of DES control, was proposed by Boole in 1854 [8].

If the plant and interface of a hybrid control system are viewed as a single component, this component behaves as a discrete event system. It is interesting to analyze a hybrid control system in this way because it allows it to be modeled as two interactive systems of discrete events that are more easily analyzed than the system in its original form [5].

The DES control object, in the case of productive systems, is generally a system composed of several elements that relate in a complex way to each other. To model it, it is considered that each of the elements that make up the control object are independent of each other and that each has its own states, which evolve in two ways: 1) depending only on the present input; 2) depending on the inputs and the states passed. Both one form and the other have evolutions (transitions) of states that depend on an event, the input signal. In addition, considering that each element of the system is independent, the evolution of the states of each component occurs asynchronously and in parallel [8]. The functions of the DES control system of the study plant are structured according to Figure 2.

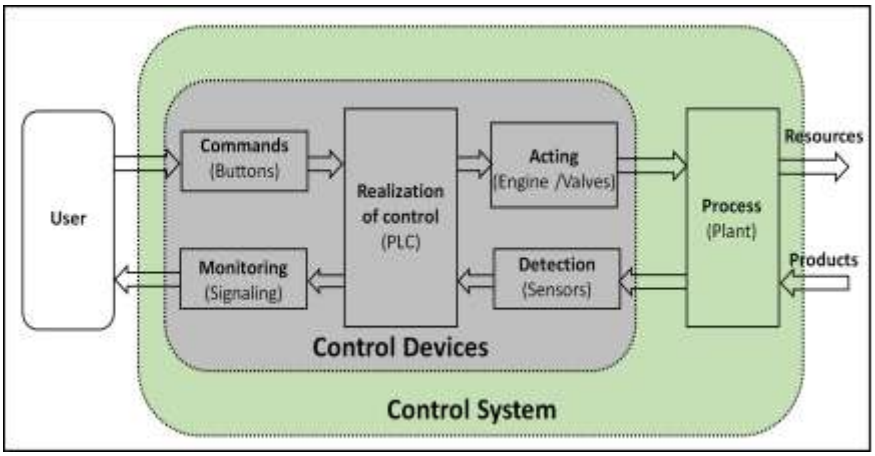

Figure 2: Conceptual diagram of the plant DES control system.

Table 2 shows all the events that occur in the tank operation process. The first column shows all stages of the process, the second column shows the discrete events of the plant, and finally in the third column there are the actions taken during the operation of the process

Table 2: Plant events

STAGES	EVENTS	ACTIONS
1	Tank 01.	Open valve 1.
2	Fill tank 01.	Turn on engine.
3	Full tank 01.	Activate MD.
4	Fill tank 02.	Open valve 2.
5	Full tank 02.	Open valve 3.
6	If tank 01 and 02 full.	Reduce engine flow.
7	Reservoir filled with valve 1 and 2 closed.	Switch off the engine.
8	Tank 01 filled with valve 1 closed.	Switch off the engine.
9	Tanks 01 and 02 filled with valve 2 closed.	Switch off the engine.
10	Engine defect.	Light signaling.

III. CONCEPTION OF THE CONTROL SYSTEM

Control theory deals with the behavior of dynamic systems, where the desired output of a system is called a reference signal [10] and [12]. When one or more output variables need to follow a certain reference over time, a controller handles the inputs of the system to obtain the desired effect on the system outputs. A PI controller was designed to control the levels of tank 1 and tank 2.These were embedded in an industrial PLC as shown in Figure 1.

PLC is used for applications involving discrete stock control in the manufacturing process industries. The availability of PLCs with features such as devices, machines, and process operations by implementing specific functions such as control logic, sequencing, time tracking, mesh control, advanced functional instructions, and improved Graphical User Interfaces (GUI) in programming these controllers, created the premises for the implementation of complex control algorithms [3] e [5]. Initially, the control system designed and presented here was tested and validated in simulation using MATLAB/SIMULINK software, then the system was embbed in the PLC of the tank plant.

3.1 Proportional-Integral-Derivative Controller (PID)

Often, simple processes can be satisfactorily controlled only through proportional action. In this case the integral and derivative actions are simply turned off. We then have:

$$u(t) = K(e(t)). \tag{11}$$

In many industrial PID controllers, instead of directly specifying the value of K, the value of the proportional band (P_b) is specified in percentage value. Note that, considering u_{max} -u_{min} = 100%, one has:

$$K = \frac{100}{P_b}. \tag{12}$$

The main function of the integral action is to make processes follow with null error, a reference signal of the jump type [9]. However, the integral action if applied in isolation tends to worsen the relative stability of the system [15]. To counterbalance this fact, the integral action is usually used in conjunction with the proportional action constituting the Integral Proportional Controller (PI), whose control signal is given by

$$u(t) = K\left(e(t) + \int_0^t g(t)\right). \tag{13}$$

The higher the value of *K*, the lower the proportional band. As seen in the study of proportional action, for a system of type 0, the higher the gain *K* the lower the value of the steady state error, but this error will never be completely annulled.

Applying the Laplace transform has the following transfer function for the PI controller

$$G_{pi}(s) = \frac{u(s)}{r(s)} = \frac{K(s + 1/T_i)}{s}. \tag{14}$$

For high values of *Ti*, the proportional action is predominant, where $Ti = \infty$ corresponds to the proportional controller. As we decrease *Ti*, the integral action begins to predominate over the proportional action, and the response tends to approach the reference more quickly.

The output of a process intuitively presents a certain "inertia" with respect to modifications in the input variable. This "inertia" is explained by the dynamics of the process that causes a change in the control variable provoke a considerable change in the output of the plant only after a certain time. Another interpretation is that, depending on the process dynamics, the control signal will be "delayed" to correct the error. This fact is responsible for transients with great amplitude and oscillation period, being able, in an extreme case, to generate unstable responses.

In the Proportional-Derivative (PD) Controller, the derivative action when combined with the proportional action has precisely the function of "anticipating" the control action in order for the process to react faster. In this case, the control signal to be applied is proportional to a prediction of the process output. The basic structure of the PD controller is given by:

$$u(t) = K\left(e(t) + T_d \frac{de(t)}{dt}\right). \tag{15}$$

Considering that *e(t + Td)* can be approximated by

$$e(t + t_d) \approx e(t) + T_d \frac{de(t)}{dt}, \tag{16}$$

One has that *u(t) ≈ Ke(t + Td),* that is, the control signal is proportional to the control error estimate *Td* units. In other words, the prediction is made by extrapolating the value of the error by the line tangent to the

curve of the error at time *t*. This predictive action tends to increase the relative stability of the system and make its transient response faster.

In practice, the gain of the derivative part in high frequencies should be limited by the addition of a *p* pole. The transfer function of the controlled PD is then given by:

$$G_{pd} = \frac{u(s)}{r(s)} = K\left(1 + \frac{spT_d}{s+p}\right) = \frac{K(1+t_d p)\left(s + \frac{p}{1+pT_d}\right)}{s+p}. \quad (17)$$

Note that the zero of the PD controller is always to the right of the pole. This setting is equivalent to that of a phase lead compensator. Note also that as we increase T_d, the zero of the controller tends to origin, meaning the predominance of the derivative action.

The PID controller combines the advantages of the PI and PD controller. The integral action is directly linked to the system precision, being responsible for the steady state error [14]. The destabilizing effect of the PI controller is counterbalanced by the derivative action that tends to increase the relative stability of the system, while making the system response faster because of its anticipatory effect [2]. The transfer function of the PID controller is given by:

$$G_{pid}(s) = \frac{u(s)}{r(s)} = K\left(1 + \frac{1}{T_i s} + \frac{spT_d}{s+p}\right) = \frac{K\left(s^2 + \frac{1+T_d T_i}{T_i}s + \frac{p+T_i p}{T_i}\right)}{s(s+p)}. \quad (18)$$

The gains of the PID controllers are adjusted to obtain the best operational performance of the processes of the plants that use this technology. Such gains are adjusted according to the process to be controlled. The proportional gain K_c provides an amplitude control action proportional to the amplitude of the input signal, the error in this case; the integration time T_i is the time required for the integral action to be doubled. The steady state error with respect to a constant value reference signal through a low frequency compensator is reduced; and the derivative time T_d is equivalent to an anticipation of the control action, considering the tendency of variation of the error. It improves the transient response through a high frequency compensator. The term derivative uses the rate of change of the error signal to introduce a prediction element into the control action.

3.1.1 PID controller embbed in the Plant PLC

The software presented in this article is applied in real plants by PID controllers embedded in a PLC using industrial instrumentation. In general, PLCs have application in the automation of discrete processes (ON-OFF control) and in the automation of continuous processes (mesh control).

3.1.2 PID Instruction parameters

The PID instruction is usually positioned on an always true line. When the line is false, the output remains at its last value. During programming, one must enter the addresses of the Control Block and Process Variables after placing the PID instruction on a line:

- **Control Block -** File that stores the data needed to operate the instruction. The length of the Control Block file is fixed in 23 words. The address of the Control Block should be entered as a file address of integer type N [ ]. For example, by entering N7: 2, elements from N7: 2 to N7: 24 will be allocated automatically;
- **Process Variable (PV) -** Element address that stores the value of the process input. This address can be the word address of the analog input where the value of the A/D input is stored. This value can also be an integer value when one wants to work with the scale input value in the range of 0 to 16383;
- **Control Variable (CV) -** Element that stores the output of the PID instruction. The range of the output value is from 0 to 16383, with 16383 being 100% of the value. This is normally an integer value, so that the PID output range can be scaled to the specific analog range required in the application.

Figure 4 illustrates the PID instruction with real addresses of the plant being studied and its parameters embedded in the PLC.

Figure 4: PID Controller Embedded on the Plant's PLC.

After entering the addresses of the Control Block, Process Variable and Control Variable, the PID SETUP - PID window is displayed, as shown in Figure 5, where the settings of the controller gain parameters designed and implemented in this work are made on Rslogix 5000 software in PLC from the plant as shown in the ladder program line of Figure 4.

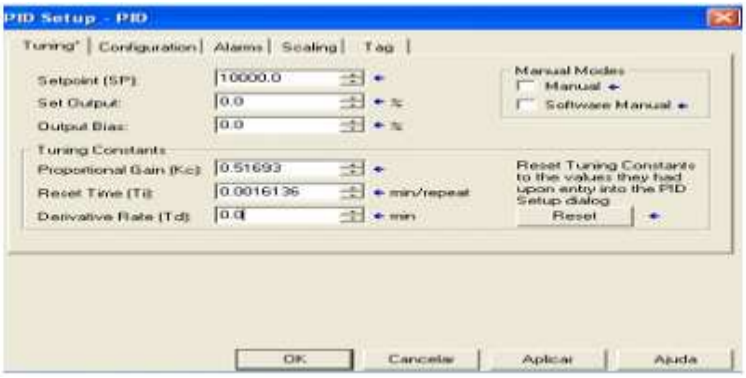

Figure 5: Adjustment of PI Parameters in the PLC of the Tank System Plant.

3.1.3 Pide Instruction

The PIDE instruction uses a speed form algorithm of the PID equation. Essentially, this means that the loop works in error change to change the output. The traditional PID algorithms used in PLCs have used positional form algorithms. A positional form algorithm works directly on the error. Although this is acceptable for simple applications, the velocity shape algorithm is much easier to apply for more advanced applications such as adaptive gains or multiloop selection [13].

Both the positional form and a speed-form PID algorithm work identically in response to a change in error. In fact, one form of the equation can easily be derived in the other [13].

The positional PID algorithm is represented mathematically by Equation (19)

$$CV = K_p E + \sum K_I E \Delta t + K_D \frac{\Delta E}{\Delta t}. \quad (19)$$

And the velocity-form PID algorithm is represented mathematically in Equation (20).

$$CV = CV_{n-1} + K_p \Delta E + K_I \Delta t + K_D \frac{E_n - 2E_{n-1} + E_{n-2}}{\Delta t}, \quad (20)$$

where $CV =$ control variable, $E =$ error, $\Delta t =$ variation of time, $K_P =$ proportional gain, $K_I =$ integral gain and K_D = derivative gain [13].

The two main differences between the forms of the PID algorithm are that the proportional term acts on the error change (ΔE) in velocity form and error (E) in the positional form. While the accumulation of the integral term is contained in the previous output (CV_{n-1}) in velocity form and in the sum of the integral term in the positional form [13].The PIDE instruction also supports two different speed shape algorithms: independent gains and dependent gains [13].

In the independent gain form, each term of the algorithm, proportional, integral and derivative, has a separate gain. If one changes a gain, it affects only that term and not any of the others [13]

$$CV = CV_{n-1} + K_p \Delta E + \frac{K_I}{60} E \Delta t + 60 K_D \frac{E_n - 2E_{n-1} + E_{n-2}}{\Delta t}. \quad (21)$$

While the algorithm in the form of dependent gain, the proportional gain is effectively changed in a gain of the controller. By changing the controller gain, the action of all three terms, proportional, integral, and derivative, are changed at the same time [13].

$$CV = CV_{n-1} + \left(\Delta E + \frac{1}{60} + 60T_D \frac{E_n - 2E_{n-1} + E_{n-2}}{\Delta t} \right). \quad (22)$$

When one uses the PIDE instruction with the determined DependIndepend parameter, the PGain, IGain, and DGain parameters are used to represent K_P, K_I and K_D. When DependIndepend is defined, the following parameters are used: PGain, IGain, and DGain to represent K_C, T_I and T_D.

The Equations (22) and (23) of PIDE are representative of the algorithms used by this instruction. The change of error values can be substituted by the change in PV (in percentage of extension) for the proportional and derivative terms, manipulating the parameters PVEProporcional and PVEDerivative. By default, the PIDE instruction actually uses the error change for the proportional term and the change in *PV* for the derivative term. This eliminates large derivative peaks at setpoint changes [13].

One can convert the gains used between the independent and dependent algorithms of the PIDE instruction using the following equations

$$K_p = K_C, \quad (23)$$

$$K_I = \frac{K_C}{T_I}, \quad (24)$$

$$K_D = K_C T_D. \quad (25)$$

Both the independent and the dependent algorithm may have controls with very good performances, provided the gains are adjusted properly. That depends on which style of PID algorithm one has more familiarity with. Some prefer independent gain style as long as they can handle individual gains without affecting other terms. Others prefer the dependent gain style, since they can, at least to some extent, change only the controller gain and cause a general change in PID loop aggressiveness without changing each gain separately [13].

The PIDE instruction provides additional features through the use of many different modes of control. In addition to traditional modes such as auto and manual, the PIDE instruction also supports the Program / Operator control concept to define who is allowed to make changes to the loop. If the loop is in program control, the user program can loop in the appropriate mode (e.g., Auto/Manual) and change the manual set point or loop output. On the other hand, if the loop is in the Operator control, the operator can change modes and values [13]. The supported control types and loop modes are:

- **Program Control -** When in Program Control, the loop mode is determined by the user program. The user program can also change the setpoint and manual loop output;
- **Operator Control -** When in the Control Operator, the loop mode is determined by the operator. The operator can also change the setpoint and manual loop output;
- **Cascade/Ratio Mode -** When in Cascade / Ratio Mode, the loop will automatically regulate its output as in Auto mode, but the setpoint will come from an external source, as connected to the SPCascade input parameter. If the UseRatio parameter is set, the SPCascade input will also be multiplied by a ratio value before being used as the setpoint;
- **Auto Mode -** When in Auto Mode, the loop will automatically regulate its output to maintain *PV* at the setpoint;
- **Manual Mode -** When in Manual Mode, the loop will set its output equal to the *CV* value entered by the user program (when Program Control) or by the operator (when in the Operator Control);
- **Override Mode -** When in Override Mode, the loop will set its output equal to the CV value set in the CVOverride parameter. Override mode is normally used for interlock conditions;
- **Hand Mode -** When in Hand Mode, the *CV* value is set equal to the HandFB parameter. The HandFB is intended to come from a hard way from a hand/auto station. When the loop is placed in the Hand Mode (it defines the parameter ProgHandReq), it indicates that the hand/auto station has ignored the control system and is directly controlling the final control element. Setting the value of CV equal to the value of HandFB (the output of the hand/auto station), the loop iterates less and can return to Auto or Manual Mode as long as it is out of Hand Mode.

Mode changes are initiated by setting the appropriate mode request parameters of the PIDE instruction. These mode request parameters are prefixed by "Prog" to indicate that it is a program request or "Oper" to indicate that

it is a request generated by the operator. For example, OperAutoReq is an operator's request to enter Auto mode [13].

The Program/Operator control states can be used to lock the PIDE instruction in the appropriate control state when necessary. For example, an automated boot sequence can be used in an application where the user program needs to block the PIDE instruction in program control to ensure that the operator does not interfere with initialization. This can be done by configuring the parameter ProgProgReq (programming request to go to program control state) [13].

IV. SIMULATED EXPERIMENTS WITH REAL DATA

In this section the results obtained in simulation with the actual plant data are presented, as shown in Figure 1. Initially, the tank system simulation was performed using MATLAB/Simulink, represented by the block diagram of Figures 6 and 7. The first step for the development of the PI controller was the identification of the Process Variables (PV) and Control Variables (VC). For the tank level control system, a PI controller was designed to keep tanks 1 and 2 full. As inputs to the control system, the error was adopted, this being the difference between the reference input and the outflow from tank 1.

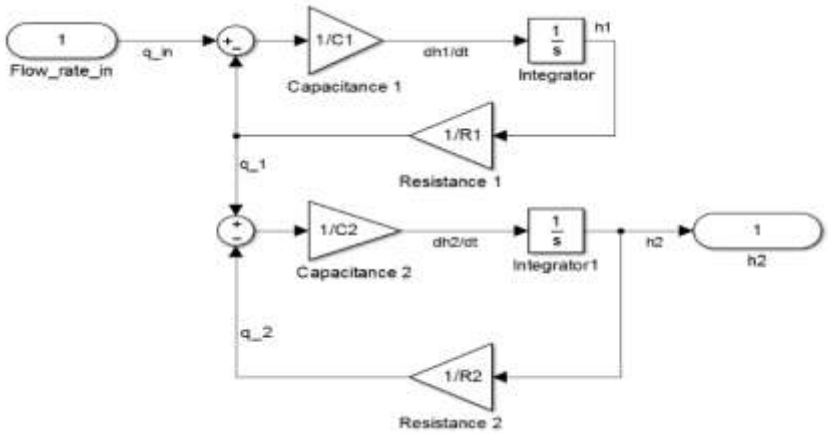

Figure 6: Block Diagram *H(s)* of the Tank Plant.

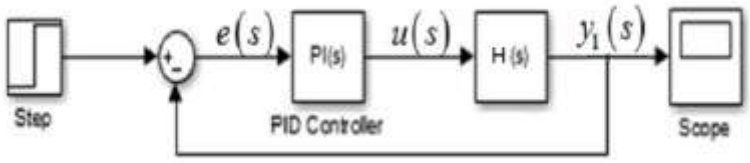

Figure 7: Block Diagram of the Compact Tanks.

A reprsentação matemática do controlador PI está apresentada na Equação (26)

$$PI(s) = P + I\frac{1}{s}, \tag{26}$$

where P = proportional gain and I = integrative gain.

Substituting in Equation (26) for the gain adjustment parameters, one has

$$0.51693 + 0.00016136\frac{1}{s} \tag{27}$$

Figure 8 shows the behavior of the system with the controller, represented by the solid line. It is observed that the system is already stable, even before the implementation of the PI controller, but after the PI controller is implemented, the system has a small oscillation, resulting in an overshoot of 8.04% in 568s and stabilizes in the setpoint in 1580s.

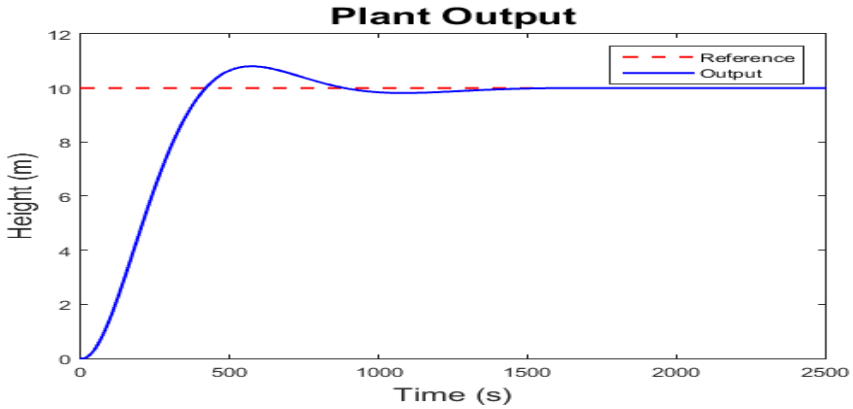

Figure 8: Step Response of the System With the Controller

Figure 9 shows the illustration of the control effort for the implemented *PI* controller, this is represented by the solid line. It is observed that the value of the control effort reached 1.01 (1%) of the setpoint value being well below the design value which is 30%.

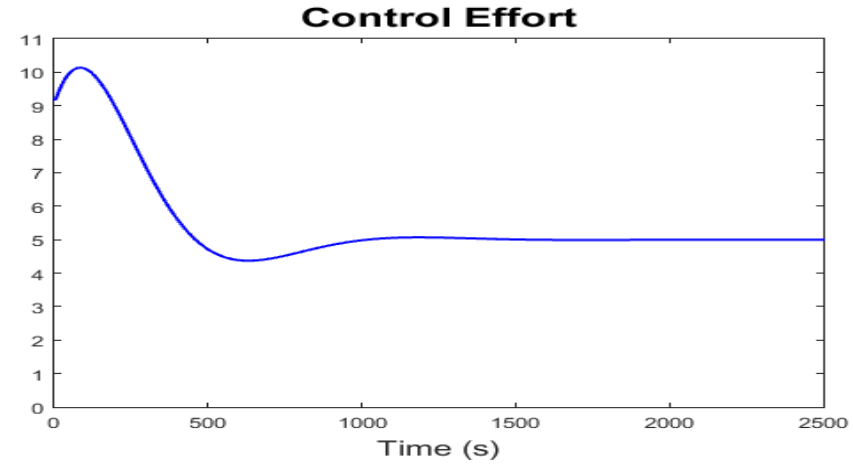

Figure 9: Control Effort of the System *PI* Controller

The error is the comparison of the reference signal with the system output signal. It can be observed in Figure 10 that initially the error signal value is very high, the opposite of the behavior of the output signal represented in Figure 8. That occurs because at the start the tanks are empty, and the PI controller was designed to keep the tanks full. When that happens, the reference signal value is equal to the value of the system output, making the error null ensuring the controlled system. This shows the good performance of the implemented PI controller.

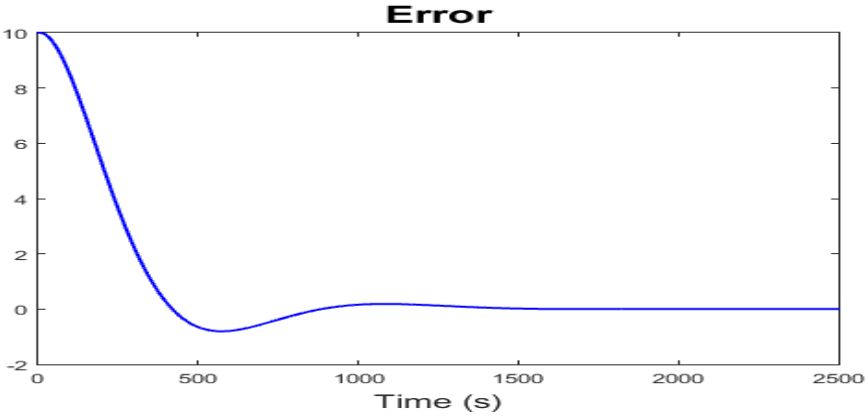

Figure 10: System error

V. CONCLUSION

The PI controller design developed and implemented in the PLC of the plant under study, obtained good results. In observing the curves of the simulated results in MATLAB/SIMULINK, the following can be observed: the system had a rapid response with overshoot below 10%. The effect of PI control effort was much lower than the pump supported and behaved very well after being disturbed, that is, it returned to the control regime very fast. Therefore, the control system implemented for the tank level control had a satisfactory result, meeting the specified design conditions.

ACKNOWLEDGEMENTS

We are thankful to the Federal University of Maranhão for the theoretical and practical teachings in the conception of the present work. We also thank the State University of Maranhão for supporting this research, the Vale SA Company for making its experts available for practical guidance in validating this work, and finally the JAV Company for guidance in the instrumentation of the plant and for guiding the development of the embedded PID controller In the PLC.

REFERENCES

Journal Papers:

[1]. Aguirre-Ghiso, J. A. (2007). Models, mechanisms and clinical evidence for cancer dormancy, Nature Reviews Cancer 7(11): 834–846.

[2]. Astrom, K. J. and Hagglund, T. (1995). Pid control theory, design and tuning, Instrument Society of America, Research Triangle ParN, NC.

[3]. Duka, A.-V. (2012). Plc implementation of a fuzzy system, The International Conference Interdisciplinarity in Engineering INTER-ENG, Editura Universitatii"Petru Maior"din Tirgu Mures, p. 317.

[4]. Skogestad, S. (2003). Simple analytic rules for model reduction and pid controller tuning, Journal of process control 13(4): 291–309.

[5]. Stiver, James A., and Panos J. Antsaklis. "Modeling and analysis of hybrid control systems." Decision and Control, 1992., Proceedings of the 31st IEEE Conference on. IEEE, 1992.

Books:

[6]. de Camargo, V. L. A. and Franchi, C. M. (2008). Controladores lógicos programáveis sistemas discretos, Editora Érica. 2ª Edição.

[7]. de Campos, M. C. M. M. and Teixeira, H. C. (2006). Controles típicos de equipamentos e processos industriais, Edgard Blücher

[8]. de Campos, M. M. and Saito, K. (2004). Sistemas inteligentes em controle e automação de processos, Ciência Moderna.

[9]. Miyagi, P.E.(2001). Controle programável: fundamentos do controle de sistemas a eventos discretos, Edgard Blucher.

[10]. Nise, Norman S. Control Systems Engineering, (With CD). John Wiley & Sons, 2007.

[11]. Ogata, Katsuhiko, and Yanjuan Yang. "Modern control engineering." (1970): 1.

[12]. Ogata, K. (2001). Modern control engineering, Prentice Hall PTR.

[13]. Franklin, G. F., J. D. Powell, and Abbas Emami-Naeini. Sistemas de Controle para Engenharia. Bookman Editora, 2013.

[14]. Franklin, G. F., Powell, J. D. and Emami-Naeini, A. (2013). Sistemas de Controle para Engenharia, Bookman Editora.

[15]. Rockwell (2016). Perform common process loop control algorithms - using the pide instruction.

Theses:

[16]. Maia, M. R. (1993). Controlador PID-AA: desenvolvimento de novas técnicas de pre-ajuste e de compensação de atroso de transporte, PhD thesis, Universidade Federal de Santa Catarina. Centro Tecnologico.

[17]. Cologni, M. A. et al. (2008). Estudo e avaliação de metodologias de auto-sintonia de controladores pid visando uma implementação em controlador industrial.