# JAVA IMPLEMENTATION OF AN INTELLIGENT SYSTEM FOR SELECTION OF CONTROLLERS USING THE QLEARNING ALGORITHM.

**Márcio Emanuel Ugulino de Araújo Júnior, marcioemanuel@ifpb.edu.br**
**Rafaelle de Aguiar Correia Feliciano, rafaelle.feliciano@ifpb.edu.br**
Federal Institute of Education, Science and Technology of Paraíba (IFPB)
Av. 1º de Maio, 720 – Jaguaribe – João Pessoa - PB
CEP 58015-430

**Fábio Meneghetti Ugulino de Araújo, meneghet@dca.ufrn.br**
**Danilo Mikael Costa Barros, danilocosta14@gmail.com**
Federal University of Rio Grande do Norte (UFRN)
Campus Universitário
Sala 72 - Centro de Tecnologia
Natal - RN
CEP 59072-970

***Abstract.*** *Currently PID controllers are widely used in industrial process control, but controllers which use AI techniques, such as fuzzy, neural networks and hybrid techniques, are no longer restricted to academic research. They are also gaining ground among industrial applications. However, each type of controller, depending on the tuning or training technique, presents better performance in certain regions or operation points, and that the design and implementation of these controllers frequently depends of proprietary software, which usually has a very high cost. Aiming to have an open source environment capable of tuning PI, PD or PID fuzzy controllers, while determining the best control signal for each instant and switching between those controllers, this work proposes an algorithm of reinforcement learning, named Q-learning to identify which controller is best suited to work in different linear regions of the system. It is used a soft-switching technique to switch between control signals and the integral of the error as metric for selection of the most suitable controller. For validation and testing of this proposal, it is used a system level control of Quanser, where the Q-learning algorithm is trained using the ε-greedy policy. Test results display the system switching among controllers and the comparison with each controller performing separately in order to prove the functioning and effectiveness of the proposed implementation.*

***Keywords**: Intelligent system, PID, Fuzzy, Q-learning, control signal soft-switching.*

## 1. INTRODUCTION

PID controllers are extensively used to control a wide variety of industrial processes, mainly because they are robust, easily understood and capable of satisfactory performance (Wang, 2001; Ingimundarson and Hägglund, 2002; Piazzi and Visioli, 2002; Zhong and Li, 2002; Chen and Seborg, 2003; Åström and Hägglund, 2004; Faccin, 2004, Fonseca *et al*., 2004). However, complexity and nonlinearities of real systems have stimulated new control techniques to emerge lately. Fuzzy controllers are suggested as one of them to replace the traditional PID ones. Controllers with fuzzy logic have been used to control different types of systems, ranging from the simplest systems, such as control of inverted pendulum shown in a study of Mohanlal and Kaimal (2002) to more complex systems such as autonomous robot navigation, as proposed in the work of Sanchez-Solan *et al*. (2007), Baturone et al. (2008) and Raguraman (2009).

In the industry the use of different controllers for the same process is already usual, but the most common technique is the gain-scheduled, as in the work presented by Fontes *et al*. (2008) which used it to control a pH neutralization process. The purpose of the gain-scheduled technique is to switch among multiple controllers of the same type (PID), tuned to act differently in the various regions of process operation.

Research has been proposed with the intention of better controlling industrial processes, considering the different characteristics inherent to classical (PID) and intelligent controllers (fuzzy), and different tunings approaches for each of these controllers in order to work in different regions of operation of the same industrial process. One of the papers regarding switching techniques among different controllers is presented by Diniz *et al.* (2010). It proposes the implementation of a system for automatic switching among controllers, PI, fuzzy and neural, implemented with the help of MatLab ® being used to control a system of levels of Quanser ®.

The idea proposed by Diniz *et al*. (2010) formed the basis for the development of this research, although no soft switching technique was proposed in the cited work. Switching technique among controllers can cause serious mechanical or electrical problems in real systems, when low signal values are applied by controllers and it is necessary to switch to higher ones provided by more aggressive controllers.

Another improvement in the work proposed by Diniz *et al.* (2010) presented in this work is the change in calculating the area of the figure when occurs the intersection of time sampling of the training process with the passage by desired reference, in this case the figure approximate to calculate of the error will be one or two triangles instead of a trapeze.

It's thinking about these improvements and the dependence of proprietary software for the design of fuzzy and PID controllers to act in the same process which this paper proposes the construction of an open source environment, able to tune PID and fuzzy controllers and able to making the selection and soft switching to the best controller, using as comparison metric the integral of the error, the system will be previously trained in a real environment, using the algorithm of reinforcement learning Q-Learning with e-greedy policy for decision making .

As a test scenario will be used a Quanser ® level system connected to a server via a power amplifier module UPM 2405 and a data acquisition board, both Quanser®. This plant is already ready and connected, as well as implementation of a service running on this server for communication via socket with the notebook where the proposed system is running.

## 2. Q-LEARNING

According to Sutton and Barto (2002), the reinforcement learning is an approach to artificial intelligence that allows an entity to learn from its interaction with the environment, through knowledge about the state of the individual in the environment, of their actions within this environment and the state changes that happened after the actions performed. An apprentice agent seeks to maximize the performance measure based on receiving reinforcements when interacting with the unfamiliar environment. Its use is recommended when there are no models, or when it is not able to get appropriate examples of situations which the learner agent will confront. The intelligent agent is an autonomous computational entity which acts without the intervention of another system and is never advised of what action to take. It aims to learn autonomously an optimal policy of action, through interaction with the environment, through trial and error.

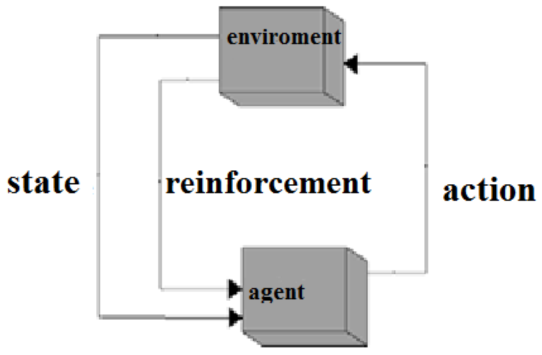


Figure 1. Representation of reinforcement learning (Sutton and Barto, 2002).

The main elements of reinforcement learning are:

- The environment where the agent will act;
- Policy (π) which represents the behavior that the system will follow to get the goal;
- Reinforcement $r_{t+1}$which is the signal given by the environment to the agent, when an action has been made and a state transition has occurred;
- Return ($R_t$) is the sum of reinforcements received over the time;
- Function of state-action value (Q (s, a)), used to evaluate the quality of the actions taken by the agent, representing an estimate of the expected total return

The purpose of the learning process is to guide the agent to take decisions (actions) which will maximize (or minimize) the expected return value. Thus callback function is defined as a sum of reinforcements over time. In simple cases, where the environment ends naturally in some specific episode (finite time), the return is given by:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_{t+n} \qquad (1)$$

But in cases where this happens (t = ∞), the return will also tend to infinity. To solve this, a discount factor is used, set in the range $0 \leq \gamma \leq 1$ to ensure that the return is finite, i.e.:

$$R_t = \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1} \tag{2}$$

To achieve this maximized return value, the agent must follow a given policy (π), which occurs in the mapping of states (s) and actions (a) in a value π (s, a) in order to converge to a policy π ^ * (s, a) which leads to solving the problem in an optimal way.

The function of state-action value Q (s, a) is used to evaluate the actions taken by agent in the state $s_t = s$ (actual state) when a action $a_t = a$ is chosen from there by following a policy π any, according the following expression:

$$Q^{\pi}(s,a) = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_t + k + 1 |\, S_t = S, a_t = a\} \tag{3}$$

To make the agent capable of making the right decisions and choose the most appropriate actions for improving the system, we use the algorithm of reinforcement learning Q-learning, one of the most important contributions to the area, proposed by Watkins et al. (1992). Among its advantages, one which stands out is the fact that it converges to optimal values of Q (s, a) without depending of the policy which is being used, i.e., the action-value function Q directly approximates the optimal value function Q * (s, a), through updates of the state-action pairs, which are made when these pairs are visited, making thus the Q-learning an off-policy method. The expression of updating the matrix of Q-values in Q-learning algorithm based on the action-value function and is denoted by:

$$Q(s\,,a) = Q(s_t\ ,a_t\,) + \alpha[r_{r+1} + \gamma \cdot max_a Q(s_{t+1}\,, a_t) - Q(s_t\,, a_t)] \tag{4}$$

where $s_t$ is the actual state, $a_t$ is the action realized in the state $s_t$, and $r_{t+1}$ the reinforcement signal received after execute $a_t$ in $s_t$, $s_t + 1$ is the next state, γ is the discount factor (0 ≤ γ < 1) and ∝ is the coefficient of learning (0 ≤ ∝< 1).

For the convergence of the algorithm is guaranteed, all state-action pair must be visited and updated continuously and can be used any policy that ensures that each state-action pair is visited often. A very common strategy is the random exploration, ε-greedy, where the agent performs the action according to the random probability ε, i.e.:

where,

$$\pi(s) = \begin{cases} a^*, & with\ probability\ 1-\varepsilon \\ a_{aleatório}, & with\ probability\ \varepsilon \end{cases} \tag{5}$$

where,

$$a^*(s) = argmax\ Q(s,a) \tag{6}$$

To carry out this work, it was decided that states would be admissible values of the error in the tank, from -4 to +4 volts, discretized with a resolution of 0.2 volts, building a table with 41 discrete states. Thus, the mapping of these states is given by:

$$S(k) = e( k ) *5 + 21 \tag{7}$$

where, S (k) is the state e (k) is the error between the desired and measured at the bottom of the tank system at time t.

In the work proposed by Diniz *et al.* (2002), the reward is given by minimizing the error using calculating the area of the error at time instant k and k +1 being approximated by a trapezoid function, as shown in Figure 2, and whose reward function suitable for the trapeze is given by:

$$R_k = \frac{|e(k)+e(k+1)|}{2} \cdot ta \tag{8}$$

being,

$$e(t) = r(t) - y(t) \tag{9}$$

where r (k) is the reference for the level of tank 2, y (k) the value actually measured for referred tank and ta the decision period (verification of switching) whose value adopted for this work was 0,5 seconds.

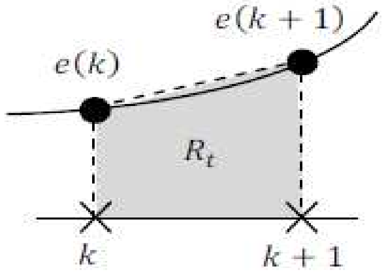


Figure 2. Area delimited by the curve of error (Diniz et. Al., 2010).

But when the moment of decision period (ta) coincides with the passage through the desired reference the approximate figure ceases to be a trapeze and becomes two triangles, as shown in Figure 3, whose functions of reward to calculate the area of triangles are given by:

$$\frac{|e(k)|}{t'} = \frac{|e(k+1)|}{t''} \tag{10}$$

$$t'' = ta - t' \tag{11}$$

$$\frac{|e(k)|}{t'} = \frac{|e(k+1)|}{ta-t'} \tag{12}$$

$$|e(k+1)| * t' = |e(k)| * ta - |e(k)| * t' \tag{13}$$

$$t' = \frac{|e(k)|*ta}{|e(k+1)|+|e(k)|} \tag{14}$$

$$t'' = ta - t' \tag{15}$$

$$A_1 = \frac{t'*|e(k)|}{2} \tag{16}$$

$$A_2 = \frac{t''*|e(k+1)|}{2} \tag{17}$$

$$A_t = A_1 + A_2 \tag{18}$$

where:
$t'$ -> Triangle’s base (time) before cross the set point.
$t''$ -> Triangle’s base (time) after cross the set point.
$ta$ -> Total sampling period (0,5s).
$A_1$ -> Total area before cross the set point.
$A_2$ -> Total area after cross the set point
$A_t$ -> Total area calculated.

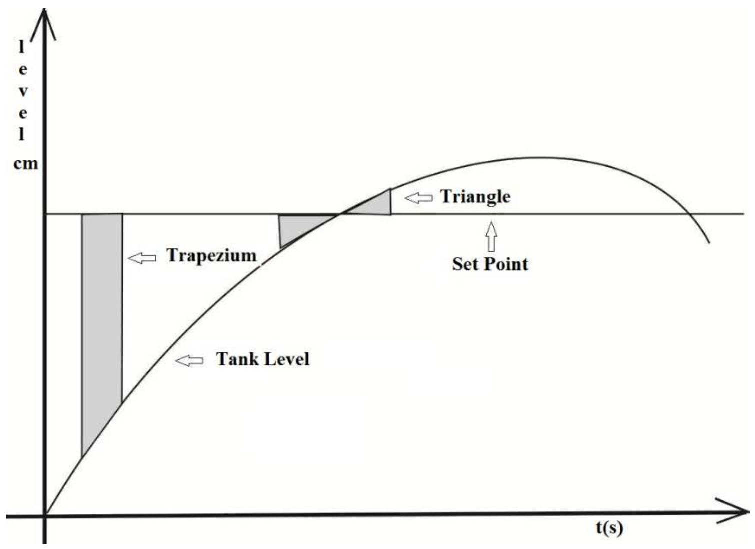


Figure 3: Area delimited by the curve of the error, crossing the reference.

The training consists of the agent's interaction with the environment in a sequence of discrete steps. At each time step t, there is a representation of the environment, a state $s_t$ (error). Based on the state $s_t$, intelligent agent selects an action $a_t$ (controller type). As a consequence of the choice of action, the environment is somehow changed, and this change is communicated to the agent through a sign of reinforcement (the error minimization function) and the shift to a new state $s_{t+1}$ of environment.

At the beginning of the learning process, the agent has no knowledge of the result of choosing between different actions, so he performs various actions and observes the results.

This interactive process allows the intelligent agent can determine, after a number of trials, the best controller to be used in every state. Thus the agent can learn an optimal policy of action that maximizes the estimated expected return represented the state-action value function Q (s, a), independent of the initial system.

It was adopted a ε-greedy policy with 80% chance of choosing the action with the highest estimate. In the first step the choice is made randomly. It was also adopted the learning coefficient $\alpha$=0.15 and the discount factor $\gamma$=1. Training was performed online, for 70 episodes, with each taking 120 periods of decision (0.5 s). The duration of each episode was approximately 1 minute. The decision period is 0.5 s the sampling period of the system is 0.1 s.

## 3. PROPOSED SOLUTION

The proposed solution is a system developed in Java using the Net Beans IDE version 6.9.1, able to implement PID controllers and fuzzy controllers Takagi-Sugeno (Takagi, 1985). These controllers can be graphically tuned through the screens of the system.

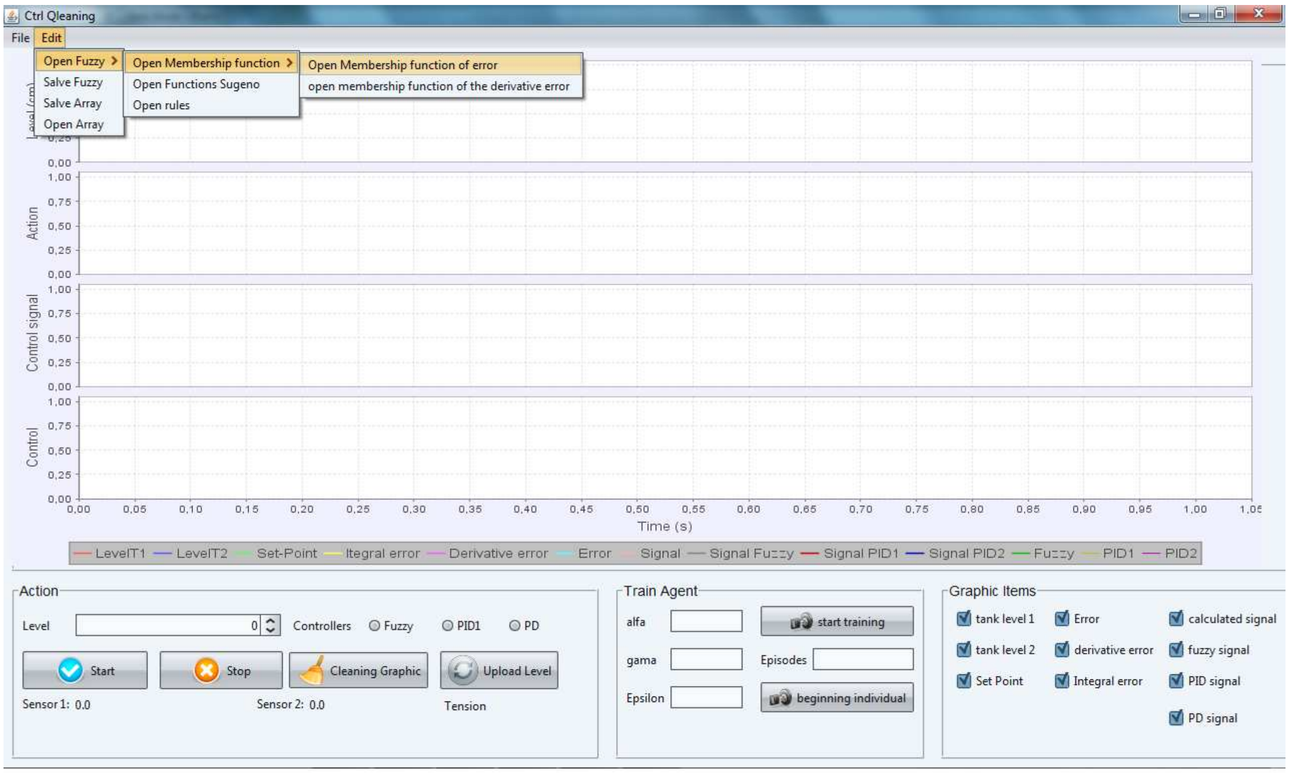
The proposed system is also able to decide what the best controller is for different operation regions of the plant, using the training algorithm Q-learning, and to check which controller had the lowest error for each sample time (ta).

For the implementation of soft switch technique among controllers it was adopted a policy of percentages, where each cycle read/write of the system, i.e., every 100 milliseconds during the switching of the controllers will be done a percentage calculation of the controller signal that was being applied and the signal to be applied It can be seen in Table 1.

Table 1: Percentages of the switching process.

| **Controller/Time** | **100 ms** | **200 ms** | **300 ms** | **400 ms** |
|---|---|---|---|---|
| Controller 1 | 80 % | 60 % | 40 % | 20 % |
| Controller 2 | 20 % | 40 % | 60 % | 80 % |

The system has many screens to configure PID controllers and fuzzy. Next, it is shown and discussed some of the screens, that can be seen in figures 4 and 5.



Level 0

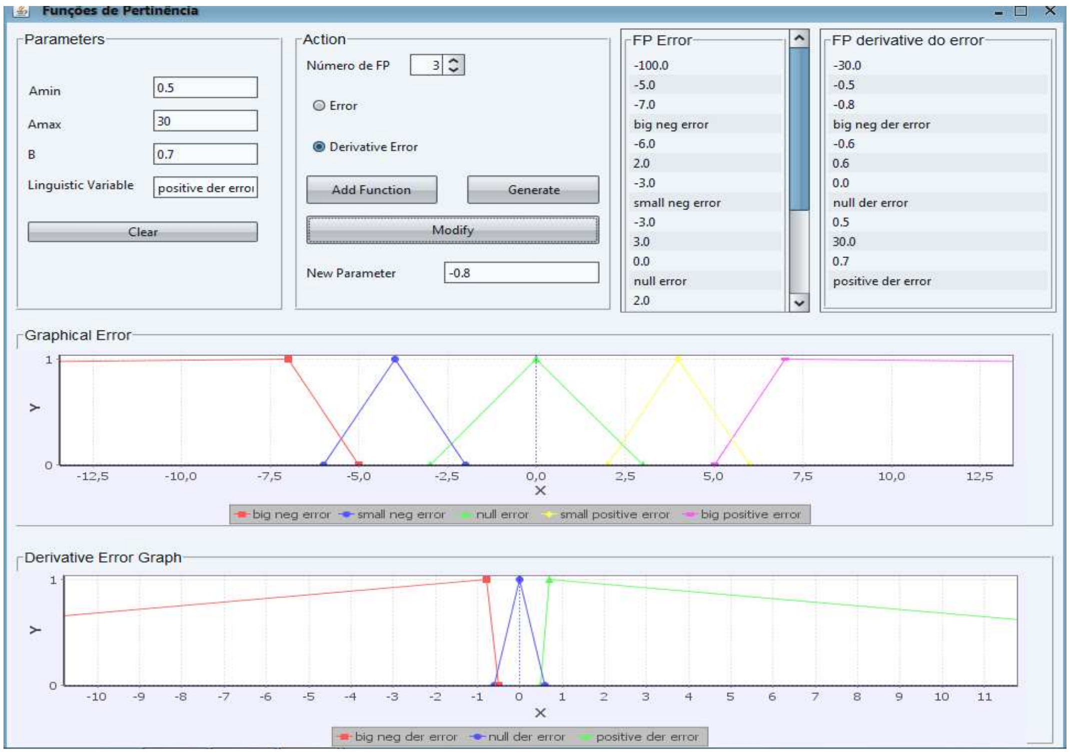
Sensor 1: 0.0

Sensor 2: 0.0

[✓ tank level 1] [✓ tank level 2] [✓ Set Point] [✓ Error] [✓ derivative error] [✓ Integral error] [✓ calculated signal] [✓ fuzzy signal] [✓ PID signal] [✓ PD signal]

Figure 4: Main Screen

The main screen of the system is responsible for showing the real-time graphical simulation of the tank system, the first graph showing the reference level, and levels of tank 1(top) and 2(lower). The second chart shows the values of error and error derivative. Third graph shows the value of controller output to be applied to the plant and the fourth graph shows what controller is in execution each instant. The main screen too permits select the desired level, and the parameters: alfa, gama, epsilon and the numbers of episodes to train Q-learning algorithm.



Amin 0.5

Amax 30

B 0.7

Linguistic Variable positive der erro

Número de FP 3

Error [✓ Derivative Error]

New Parameter -0.8

| FP Error | FP derivative do error |
|---|---|
| -100.0 | -30.0 |
| -5.0 | -0.5 |
| -7.0 | -0.8 |
| big neg error | big neg der error |
| -6.0 | -0.6 |
| 2.0 | 0.6 |
| -3.0 | 0.0 |
| small neg error | null der error |
| -3.0 | 0.5 |
| 3.0 | 30.0 |
| 0.0 | 0.7 |
| null error | positive der error |
| 2.0 | |

Figure 5: Membership Functions Screen.

In screen of membership functions we can define the number and parameters of membership functions for input variables: Derivative Error and Error.

## 4. TEST ENVIRONMENT

The chosen test environment uses a system of coupled tanks of Quanser ®, a Power Amplifier Module UPM 2405-240, a capture card and a level plant (Quanser, 2008) with two coupled tanks, level sensors and a hydraulic pump with double direction attached to the upper tank, as can be seen in figure 6.

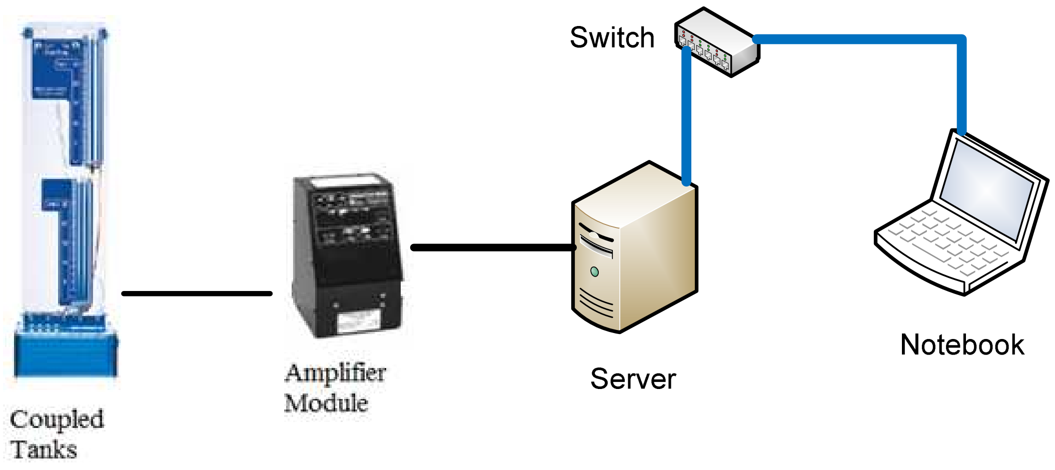


Figure 6: Test Environment.

The need to use the power amplifier module is mainly due to the low current and tension provided by capture board in to the server that is not sufficient to feed the tank system pump. The electrical and hydraulic specifications of the tank system are shown in Table 2 (Quanser, 2008).

To establish the communication between the proposed system running on the laptop and the server of the tanks, it was used a new Java class incorporated into the system. It enabled communication via socket with the service running on the server.

The communication via sockets or IP: port uses Fast Ethernet network of the lab, something that does not compromise the communication with system of tanks, knowing that this communication presents a response time of less than 1 ms and sampling rate of the system is 100 ms.

The electrical and hydraulic characteristics of tanks are shown in table 2, extracted from the tank manual (Quanser, 2008).

Table 2: The Electrical and Hydraulic Specifications of Tank System

| **Specification** | **Value** | **Unity** |
|---|---|---|
| Pump Constant Flow | 4.6 | $cm^3$/s/V |
| Pump Max Flow | 100 | $Cm^3$/s |
| Pump Voltage | ±15 | V |
| Pressure Sensor | ±12 | V |
| Pressure Range | 0 ~ 6.89 | KPa |
| Sensitivity | 5 | cm/V |

The power module UPM 2405-240 is capable of providing a continuous voltage of ± 24 volts and a maximum current of 5 Amps. Its specifications are shown in Table 3.

Table 3: UPM Power Module 2405-240, Electrical Parameters

| **Parameter** | **Value** | **Unity** |
|---|---|---|
| Amplifier Voltage Gain | 1,3 ou 5 | V/V |
| Maximum Direct Voltage Amplified | ±24 | V |
| Maximum Direct Current Amplified | 5 | A |
| Direct Voltage of Exit | ±12 | V |
| Maximum Direct Current of Exit | 1 | A |
| Alternate Current Supply | 100/120/230/240 | V |

## 6. RESULTS

The graphs presented in this work were taken from the main screen of the system running on the laptop communicating with the server system of tanks via Fast Ethernet network, as shown in Figure 7.

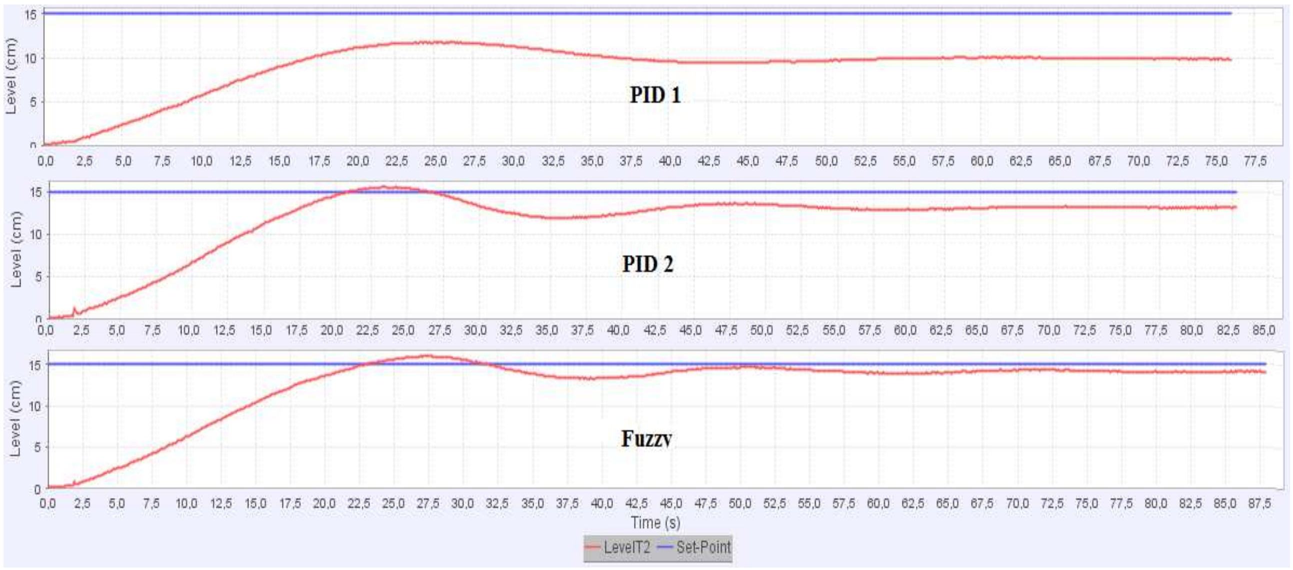


Figure 7: Graph Set Point 15 cm

Figure 7 shows the graph for the three controllers working separately, trying to keep a reference level of 15 cm. The empirical method was used to tune controllers with the purpose of forcing them to have different signals and to make clear how the process of soft switching works. It is better visualized in Figure 8.

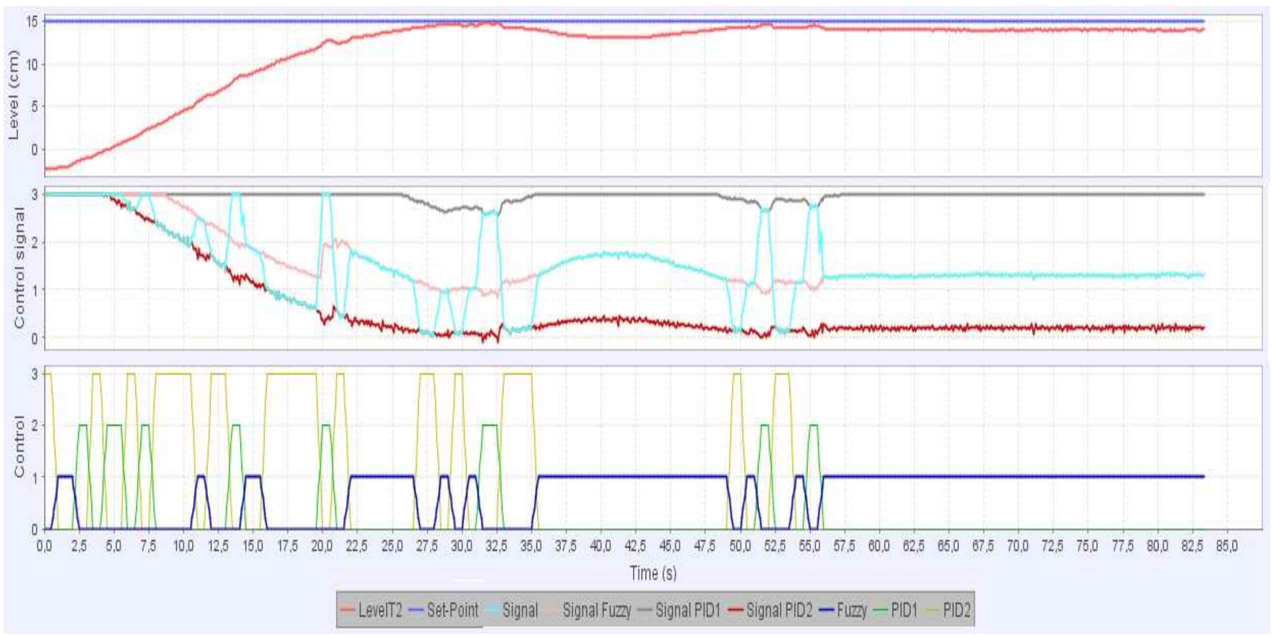


Figure 8. Graph with controller’s switch.

Figure 8 shows in the first graph the performance of controllers working together to control the same process, trying to keep a reference level of 15 cm. Controllers working together, and using the Q-learning algorithm for selecting the best one, presented better performance than when separated. Even when compared to the best controller shown in Figure 7, the fuzzy one, the proposed switching solution (graph 1 of figure 8) does not present overshoot and also shows the smallest oscillatory period.

The second graph of Figure 8 shows the signal values that each controller is calculating, but also the signal calculated in the soft switching. It demonstrates the technique of soft switching between the control signals – it is displayed by baby blue line that represents the calculated signal.

In the third graph is shown which controller is acting for each instant, becoming clear by observing the figures 7 and 8 which controller has the smallest steady state error. According to figure 7, it is the fuzzy controller and this is the one selected by Q-learning for instants of time greater than 55 seconds.

## 6. CONCLUSION

The proposed solution proved to be effective when implementing fuzzy and PID controllers, as can be seen in the graphs of Figure 7. It was able to decide what the best controller is to each time instant using the Q-learning algorithm and to make the soft switch among controllers, as seen in graph 2 of figure 8.

The switching solution presented better performance than each controller acting separately. It has minimized the settling time, oscillations and system overshoot and takes the best features of each controller. The proposed solution is feasible for tuning and making the selection of the best controller to act in the same process.

The system presented in this paper will be available as open source, so other researchers wishing to use it or improve the system will have access to the source code.

## 7. ACKNOWLEDGEMENTS

The authors would like to thank SETEC/MEC and CAPES for the initiative to promote inter-institutional master's project between Federal Institute of Science, Education and Technology of Paraíba (IFPB) and Federal University of Rio Grande do Norte (UFRN), to CNPq for the financial support to Danilo Mikael Costa Barros and to Demetrius Coutinho for his contribution on developing a new Java class for this work.

## 8. REFERENCES

Åström, K. J. and T. Hägglund (2004), Revisiting the Ziegler-Nichols Step Response Method for PID Control, Journal of Process Control, 14, 635-650.

Baturone, I., F. J. Moreno-Velo, V. Blanco and J. Ferruz (2008), Design of Embedded DSP-Based Fuzzy Controllers for Autonomous Mobile Robots, IEEE Transactions On Industrial Electronics, Vol. 55, No. 2.

Chen, D. and D. E. Seborg (2003), Design of Decentralized PI Control Systems Based on Nyquist Stability Analysis, Journal of Process Control, 13, 27-39.

Diniz A. A. R., Lemos A. J. J. F., Pires P. R. M., Kanazava S. M., Melo J. D., Dória A. D. N.(2010), Aplicação do Q-Learning para definição da política ótima de acionamento de controladores PID, Neural e Fuzzy em um processo de controle de nível., XVIII Congresso Brasileiro de Automática , Bonito-MS.

Faccin, F. (2004), Abordagem Inovadora no Projeto de Controladores PID, Dissertação de Mestrado, PPEQ/UFRS, Porto Alegre, Rio Grande do Sul, Brasil.

Fonseca, M. O., C. F. Seixas e B. S. Torres (2004), Avaliação de Desempenho e Auditoria de Malhas de Controle. InTech Brasil, 63, 32-37.

Fontes, Adhemar B.; Santos, Marcelo B.; Souza, Ramon A. R.; Achy, Acbal R. A.; Maitelli, André L.; Campos, Mário C. M. M, (2008), Técnicas de Controle Aplicadas em um Processo de Controle de pH.

Ingimundarson, A. and T. Hägglund (2002), Performance Comparison Between PID and Dead-Time Compensating Controllers, Journal of Process Control, 12, 887-895.

Mamdani, E. H. (1975), An Experiment in Linguistic Synthesis With a Fuzzy Logic Controller, Machine Studies, Vol. 7, n 1, pág. 1-13.

Mohanlal, P. P. and M. R. Kaimal (2002) , Exact Fuzzy Modeling and Optimal Control of the Inverted Pendulum on Cart, Las Vega, Nevada USA.

Piazzi, A. and A. Visioli (2002), Improved PI Control via Dynamic Inversion. In: 15th IFAC World Congress, Barcelona, Spain.

Quanser (2008), Coupled Water Tanks, Especialty Chalenge, Sheet 13-1, Review C

Quanser (n.d), Universal Power Module, User manual, Document Number 528, revision 02.

Raguraman, S. M., D. Tamilselvi and N. Shivakumar (2009), Mobile Robot Navigation Using Fuzzy logic Controller, International Conference On “Control, Automation, Communication And Energy Conservation".

Sánchez-Solano, S., A. J. Cabrera, I. Baturone, F. J. Moreno-Velo and M. Brox (2007), FPGA Implementation of Embedded Fuzzy Controllers for Robotic Applications, ieee transactions on industrial electronics, Vol. 54, N. 4.

Sutton, R. S. and Barto, A. G. (2002). Reinforcement Learning: An Introduction. 4th printing, The MIT Press.

Takagi, T. and M. Sugeno (1985), Fuzzy Identification of Systems and its Applications to Modeling and Control, IEEE Transaction Systems, 15:116–132.

Wang, Q. (2001), Auto-tuning of PID Controllers, Journal of Process Control, 11(1):105-107.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning: Machine Learning. Kluwer Academic Publishers, pp. 279-292.

Zhong, Q. and H. Li (2002), A Delay-Type PID Controller, 15th IFAC World Congress, Barcelona, Spain.

## 9. RESPONSIBILITY NOTICE