



**UNIVERSITY  
OF GÄVLE**

**FACULTY OF ENGINEERING AND SUSTAINABLE DEVELOPMENT**

# PID Control of Water in a tank

*Maria João Mortágua Rodrigues*

June 2011

**Bachelor's Thesis in Electronics**

**Bachelor's Program in Electronics/Telecommunications**

**Examiner: Niclas Björsell**

**Supervisor: Niklas Rothpfeffer**

## **Preface**

Thanks to those who helped me through these 3 years, I've worked hard but I was not the only one. And thank you to all the people involved in the ERASMUS program, which make possible for students like me to fly abroad and meet new and healthy cultures, to gain knowledge and experience.

Thank you to my parents, who raised me believing in me and appreciating me.

## **Abstract**

The thesis assignment was to build a PID control that was able to control two tanks of water. The system had to be capable of read a certain value, the value that we speak is the high of the water. There for, the system should fill the corresponding tank with water, of course, until the high that was chosen.

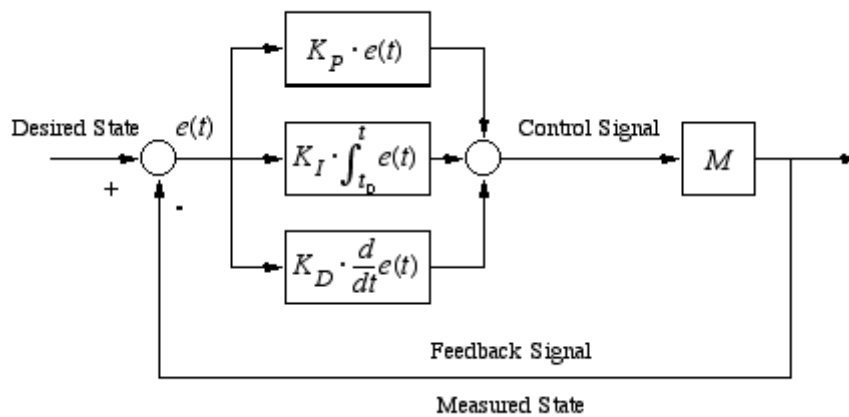
A PID control uses tree essentials values to be able to control with precision, they are usually called: P, I and D. These values can be found by applying some procedures; in this thesis two procedures were applied. So at the end, we get two values for each constant (PID). In this thesis these two values are compared in order to choose which method was the accurate.

## Table of contents

Preface.....	ii
Abstract.....	iii
Table of contents .....	iv
1 Introduction .....	5
2 Theory .....	7
2.1 Feedback Control .....	7
2.2 PID Controller .....	8
2.2.1 Proportional term.....	10
2.2.2 Integral term .....	10
2.2.3 Derivative term.....	11
2.3 Ziegler-Nichols Tuning.....	12
2.3.1 The First Method .....	12
2.3.2 Second Method .....	13
3 Process and results .....	17
3.1 Equipment .....	17
3.2 First Tuning Method.....	18
3.2.1 Upper Tank.....	18
3.2.2 Lower Tank .....	22
3.3 Second tuning Method.....	25
3.4 1st Method Vs 2 <sup>nd</sup> Method .....	29
3.5 Main LabView Control Program.....	31
4 Discussion.....	36
5 Conclusions .....	38
References .....	39
Appendix A .....	A

# 1 Introduction

This work is based on a PID control of the level of the water in a tank. The user has the power to choose a certain level of water and the system has to be capable of adjusting itself in order to maintain that certain level. To that, a LabView program was built, this program gives the user the possibility to choose the level of the water, and by constant recalculations(Fig.1), involving the error generated by the input value and the output value, the output value is adjusted in order to be closer to the desired level.



*Fig. 1 - PID Loop Control*

To do that is necessary to constantly read the input value that in this case is the current level of the water, and for that the system has the NI USB-6008 acquisition board (Fig.2), that enables several inputs and two outputs of 5V. This system has two inputs and one output, the inputs are the two readings of the level of water in the tanks, the upper and the lower tank and the single output is the voltage that the pump needs to receive in order to start working.



*Fig. 2 - NI USB-6008 Acquisition Board*

As I talked about in the beginning, this system is constituted by a PID control and by that we know the PID, has explained before; involve several calculations and several constants that affect the general response of the system. Those constant need to be found and for that we have several methods, but in this thesis they are going to be used only two of those methods. These methods involve different procedures but should, in perfect condition give the same values for the constants, but in practice that is not true, so in the results we are going to compare them e conclude which one gave a better result.

Once we are talking about two tanks, when the lower tank is being used we have to consider that the upper tank is being filled with water to, so we have to consider that the upper tank may overflow, and by that is justified the need of a warning to the possibility of the upper tank overflow.

## 2 Theory

### 2.1 Feedback Control

The success of feedback control is because this system makes everything faster, more precise and less sensitive to disturbances. The open loop control, regarding its simplicity, it's only advised in system when the outputs and inputs are known and in which there is no disturbance associated.

In system with feedback control there is a big disadvantage which is the probability of the system get unstable, for that the correct controller must be chosen, and it must be perfect for the system that is being monitored.

The basic structure of conventional feedback control systems is shown in Fig.3, using a block diagram representation. The purpose is to make the variable  $y$  follows the Set-point  $r$ . For that, the variable  $u$  is manipulated at the command of the controller. The variable  $d$  is considered as disturbances. The disturbance may be any factor that influences the process variable.

## 2.2 PID Controller

Feedback loops have been controlling continuous processes since 1700's. [2]

Today, there are several more controllers, but most of all derives from the PID controller.

“The PID controller is by far the most common control algorithm. Most feedback loops are controlled by this algorithm or minor variations of it. It is implemented in many different forms, as a stand-alone controller or as part of a DDC (Direct Digital Control) package (...). Many thousands of instrument and control engineers worldwide are using such controllers in their daily work.” [1]

A PID controller is a controller that includes the proportional element ,“*P element*”, the integral element, “*I element*” and the derivative element ,“*D element*”.

Defining  $u(t)$  as the controller output, the final form of the PID algorithm is:

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

Where:

$P_{out}$ : Proportional term of output

$K_p$ : Proportional gain, a tuning parameter

$K_i$ : Integral gain, a tuning parameter

$K_d$ : Derivative gain, a tuning parameter

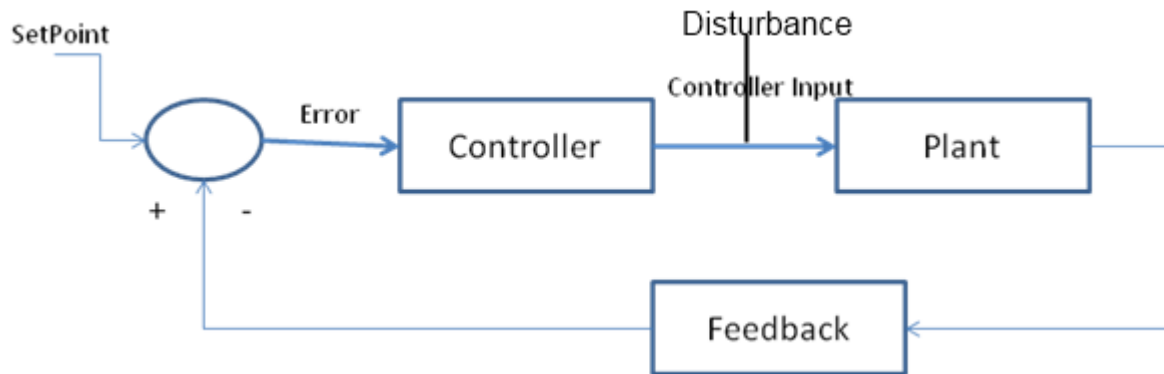
$e$ : Error =  $SP - PV$

$t$ : Time or instantaneous time (the present)

$MV$ : Manipulated variable



The figure 3 show the simple structure of a PID controller.



*Fig. 3 - PID Controller Structure*

- **Plant** – The physical parts of the system that is supposed to be controlled;
- **Feedback** – from the devices that measure the variable we want to control;
- **SetPoint** – This is a value, which is converted to a voltage that the process drives towards;
- **Error Signal** –  $\text{Error} = \text{SetPoint} - \text{Measured}$ ;
- **Disturbances** – A disturbance is all of the things that can drive a system to error, not considering the error talked above. A disturbance in the system that is talked after is for example, interference on the electrical system.
- **Controller** – The controller can be considerate the most important part of this system. It will read the SetPoint, process the error, and give the output to the Plant. It's very important that the system works correctly, and for that there are several methods of tuning the constant talked above. Those methods are being explained ahead.

### 2.2.1 Proportional term

[3] The proportional influence is proportional to the generated error.

The proportional term is given by:

$$P_{\text{out}} = K_p e(t)$$

The higher the error the higher the proportional control which is clearly seen in the equation. That conclusion leads us to another one that is that: the proportional control leads the system to a fast SetPoint. But it has a disadvantage, it has steady-state error, and that error can lead to an overshoot when the system gets to the SetPoint. One way to avoid it is to increase the proportional term, but that can lead to an unstable system.

### 2.2.2 Integral term

The integral influence is proportional to the variation of the error on time.

The integral term is given by:

$$I_{\text{out}} = K_i \int_0^t e(\tau) d\tau$$

The most important benefit is that this term eliminates the steady-state error, but it has a disadvantage which is the fact that the stability of the system is affected to. Regarding the upper equation we can conclude that this integral term depends on pass values of the error.

### 2.2.3 Derivative term

The derivative term is proportional to the rate of change of the error, as we can see on the equation below.

The derivative term is given by:

$$D_{\text{out}} = K_d \frac{d}{dt} e(t)$$

This term makes an estimation of the future error and by that it can increase or decrease the speed of correction, because it can work in an early way when there are detected any changes on the error. This term is very sensitive to disturbances.

If the derivative term only changes with the rate of change of the error, if the error do not change then we don't have derivative influence.

## 2.3 Ziegler-Nichols Tuning

### 2.3.1 The First Method

The first method is applied to plants with step responses of the form displayed in Figure 8. [4] This type of response is typical of a first order system, such as that induced by fluid flow. It is also typical of a plant made up of a series of first order systems. The response is characterized by two parameters,  $L$  the delay time and  $T$  the time constant. They can be found by drawing a tangent to the step response at its point of inflection (maximum sloop) and noting its intersections with the time axis and the steady state value, as shown in figure number 8. The plant model is therefore:

$$G(s) = \frac{K e^{-sL}}{T_s + 1}$$

Ziegler and Nichols derived the following control parameters based on this model:

PID Type	$K_p$	$T_i = K_p / K_i$	$T_d = K_d / K_p$
P	$\frac{T}{L}$	$\infty$	0
PI	$0.9 \frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2 \frac{T}{L}$	$2L$	$0.5L$

Fig. 7 - Ziegler-Nichols Recipe – First Method

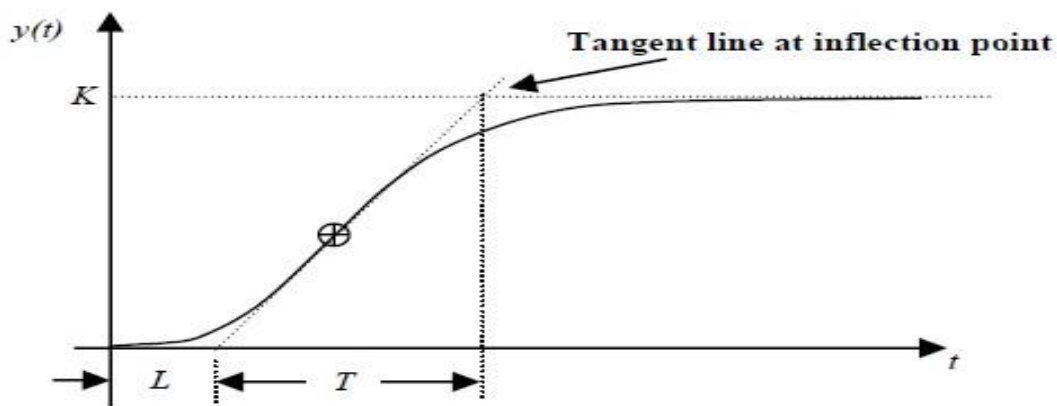


Fig. 8 - Response Curve for Ziegler-Nichols First Method

### 2.3.2 Second Method

The second method targets plants that can become unstable under proportional control. This technique has to be applied in a closed loop system because this technique requires the use of the PID controller.

The steps for tuning a PID controller via the 2<sup>nd</sup> method are as follows;

Using only proportional feedback control:

1. Reduce the integrator and derivative gains to 0;
2. Increase  $K_p$  from a low value, it may vary depending on the system, to some critical value  $K_p=K_{cr}$  at oscillations occur. If it does not occur then another method has to be applied. Note the oscillations have to be of equal period otherwise they aren't reasonable to be applied;
3. Note the value  $K_{cr}$  and the corresponding period of sustained oscillation,  $P_{cr}$ .

The controller gains are now specified as follows:

<b>PID Type</b>	<b><math>K_p</math></b>	<b><math>T_i</math></b>	<b><math>T_d</math></b>
P	$0.5 K_{cr}$	$\infty$	0
PI	$0.45 K_{cr}$	$\frac{P_{cr}}{1.2}$	0
PID	$0.6 K_{cr}$	$\frac{P_{cr}}{2}$	$\frac{P_{cr}}{8}$

*Fig. 9 - Ziegler Nichols Recipe – Second Method*

## **LabView- Laboratory Virtual Instrument Engineering Workbench**

LabVIEW is a graphical programming environment; it makes possible the readings of values, and its manipulation.

It work based on VI, Virtual Instruments, each one of them is constituted by the work environment and the block diagram. The work environment has all of the user «stuff», like graphic demonstration etc, and the block diagram is where all the programming goes, hidden from the user. The LabView can run several VI's at the same time with no interferences between them, and in each VI program it's possible to execute several tasks at the same time to.

Every created data can be storage I the computer or even printed out, or simply thrown away by the user.

LabView has a bunch of base structure, those structures the programmer cannot change, but he can use them in several different ways, those structures are called blocks.

This platform made the job much easier because it has many advantages:

- Faster Programming;
- Hardware integration with LabView;
- Advanced built-in Analysis and Signal Processing;
- Data Display and User Interfaces;
- Multiple target and OSs;
- Multiple Programming Approach;
- Data Storage and Reporting;
- Software Services, Training and Support;
- File Sharing and Collaboration with LabView Users Worldwide.

The board that it was used to acquire the signals was the NI\_USB 6008, as the follow image illustrate.



*Fig. 4 - NI-USB 6008*

Important information about the board:

- 8 analog inputs (12-bit, 10 kS/s);
- 2 analog outputs (12-bit, 150 S/s); 12 digital I/O; 32-bit counter;
- Bus-powered for high mobility; built-in signal connectivity;
- OEM version available;
- Compatible with LabVIEW, LabWindows/CVI, and Measurement Studio for Visual Studio .NET;
- NI-DAQmx driver software and NI LabVIEW SignalExpress LE interactive data-logging software.[4]

The National Instruments USB-6008 provides basic data acquisition functionality for applications such as simple data logging, portable measurements, and academic lab experiments. It's powerful enough for sophisticated measurment, other then student use.

+5 V	34	33	PFI 0
D GND	32	31	P1.3
P1.2	30	29	P1.1
P1.0	28	27	P0.7
P0.6	26	25	P0.5
P0.4	24	23	P0.3
P0.2	22	21	P0.1
P0.0	20	19	D GND
LED	18	17	D+
VBUS	16	15	D-
AI GND	14	13	AI GND
AI 4 (AI 0-)	12	11	AI 0 (AI 0+)
AI 5 (AI 1-)	10	9	AI 1 (AI 1+)
AI 6 (AI 2-)	8	7	AI 2 (AI 2+)
AI 7 (AI 3-)	6	5	AI 3 (AI 3+)
AI GND	4	3	AI GND
AO 1	2	1	AO 0

Fig. 5 - USB-6008 OEM Terminal Assignments



### 3 Process and results

#### 3.1 Equipment



Fig. 7- Both Tanks and Pump

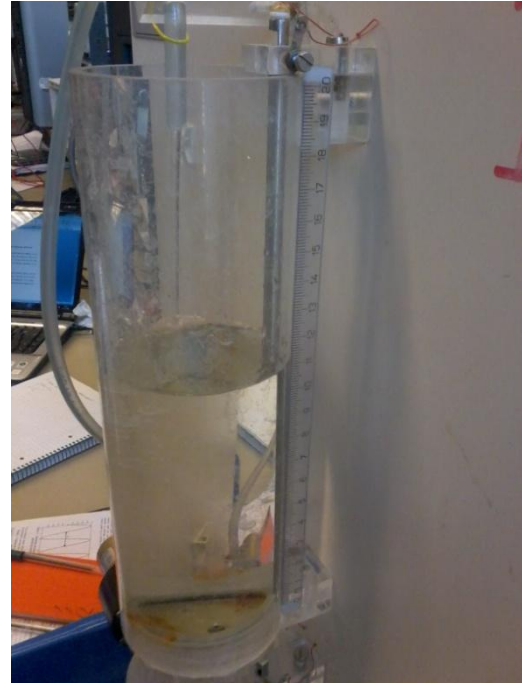


Fig. 6 - Single Tank and Ruler



Fig. 7 - Sensor Connections



Fig. 8 - Pump

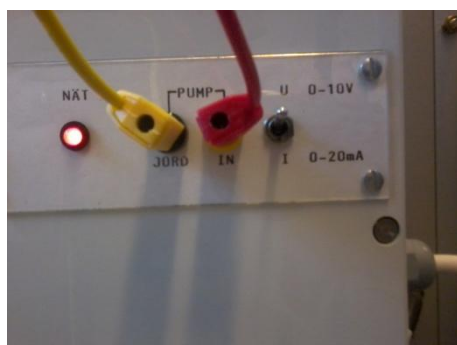


Fig. 16- Pump Connections



Fig. 9 - General Connections

### 3.2 First Tuning Method

As seen in the theory, the first tuning method implies the step response of the system. The step response can illustrate the fastest or slowest response of one system, and by that the user can get access to many important information of the system.

By that, in this method the entrance of the system was a step impulse, characterized by amplitude equal to two. There was the need of having amplitude equal to two because the system didn't respond to amplitude equal to one. After we analyze the step response of the system we can get the several values needed to calculate the  $K_p$ ,  $T_i$  and  $T_d$  values, important for our system calibration.

#### 3.2.1 Upper Tank

The next figure illustrates the LabView program built specifically to the first tuning method, for the upper tank, the board that we are going to see it's called the work environment, this board allows the user to control all of the programs at the diagram block(Fig. 20).

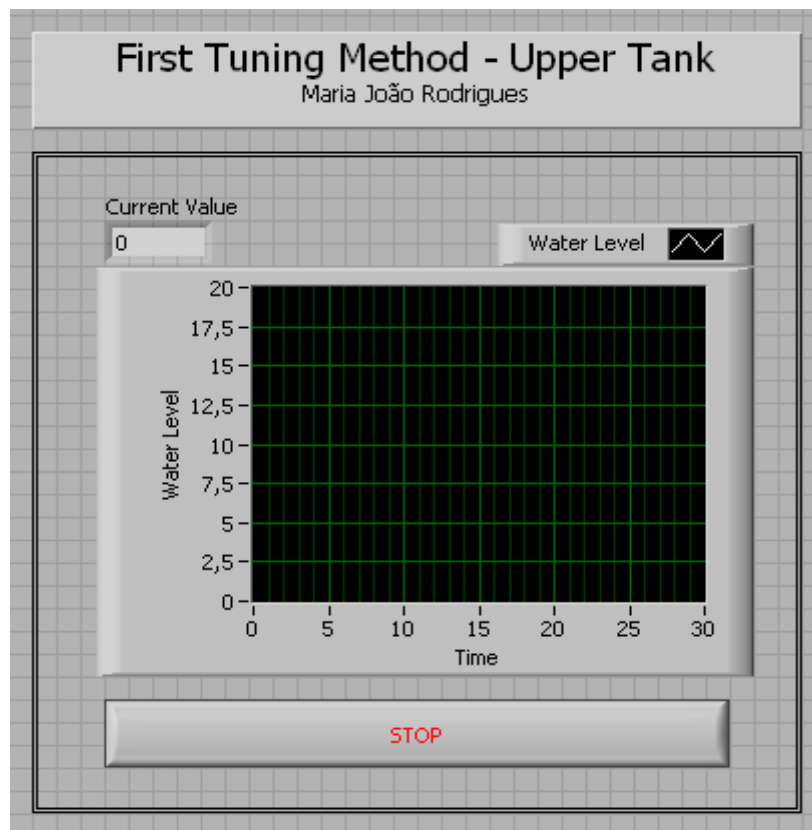


Fig. 10 - Work Environment Upper Tank

In the work environment is showed the graphic that illustrates the Water Level in order of the Time in milliseconds. It is also showed the current value, so we can keep track of the instantaneous value of the water level not only in the graphic.

The next figure is the so called Block Diagram, it represent all of the background programs that make the run of the program possible. The grey line around the blocks is a loop control, it does exactly what a loop control does, it repeats all of the commands until the STOP button is pressed.

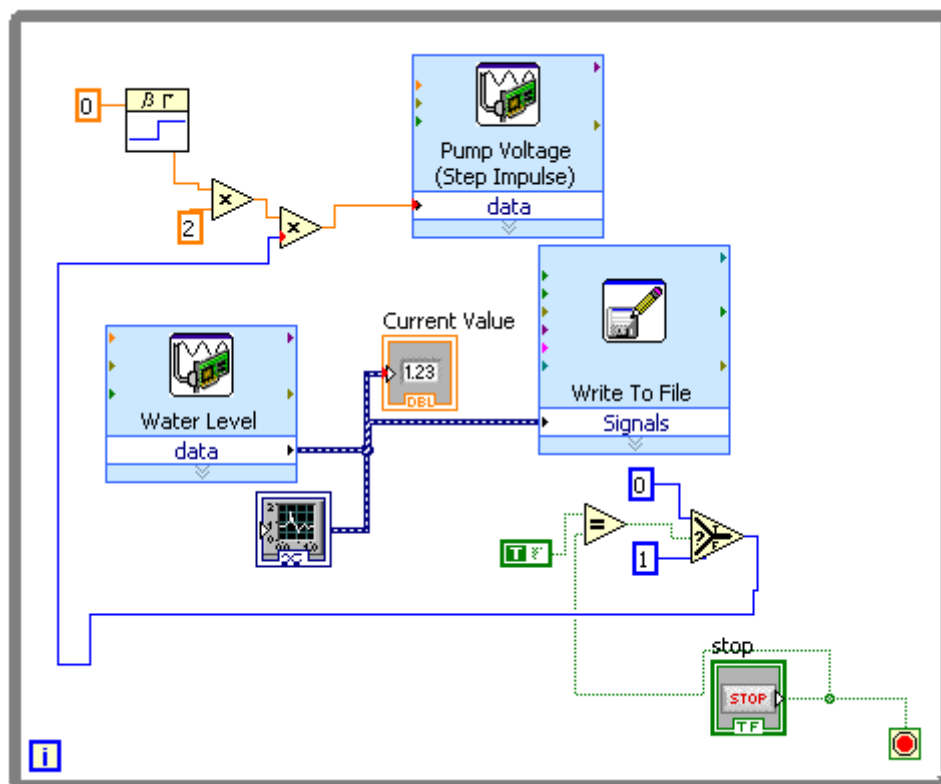


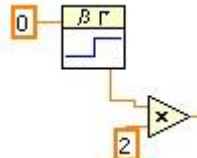
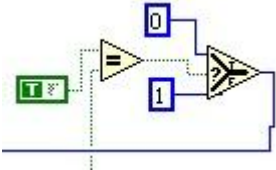
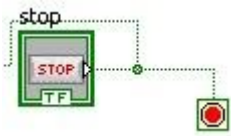




Fig. 19 - Block Diagram of the Upper Tank Tuning

In this program we have an input directly to the pump, this input corresponds to an step impulse with amplitude equal to two. At the same time the water level is being read and written into a file. Meanwhile there is an “if sequence” that checks if the STOP button was pressed, if it was, that means the first condition is true, and then the output is multiplied by 0, that means that the pump will stop working.

In the next page, at Table 1, all blocks are explained individually to better comprehension, there are several more print outs that show a lot of important information about the connection “behind” of some blocks, therefore they are in the Appendix A, in order to get a nice organization.

Table 1 - Individual Blocks Upper tank Tuning

	<p>This block is the correspondent to the Graphic seen in the Fig. 19. This is called an input block because it only receive information and do not have an output. Naturally because a normal graphic only show information.</p>
	<p>This block allows the communication between the LabView and the board. This specific block is connected to the step impulse that means the board is going to respond to a step impulse entrance. That entrance takes place in the <i>data</i> connection which is called an input. The output of this block is the voltage going directly to the pump.</p>
	<p>This is the structure of a step impulse; the need of a bigger step was because the system did not respond to a simple step impulse. Therefore was added a multiplied constant to the signal, that constant was 2, and it was good enough to make possible a system response.</p>
	<p>This structure serves the purpose of stopping the pump from working. It checks the state of the button STOP, and if it's true, that means that the button was enabled, and therefore the pump need to be stopped. So if it's that case, the answer for the first equal will be true, so the input of the <i>if question</i> will be true, and therefore the output of it will be the constant wired to the <i>true half</i> (the upper half) and not the answer wired to the <i>false half</i> (the lower half). That specific answer will be multiplied by the input of the pump which means if it was 1, the program stays the same, and if it was 0, the pump will turn off.</p>
	<p>This is the loop control STOP button. It's a predefined button that appears when a loop control is created. It enables the end of the program whenever the user feels like. The button is wired to a red square, that square is the real program stopper, and if it receives a input <i>true</i> it will stop the loop, therefore when the user clicks on the button one time, the button change its value to <i>true</i> to make possible the end of the program.</p>
	<p>This block could look like the second block presented on this list, but this block is not an output but an input. This block reads the voltage coming from the sensor that measure the high of the water in the tank. Therefore is an important part of this program, because the high of level is an essential data for the PID control. In this case the <i>data</i> is an input because it is giving the system some information about the outside values.</p>
	<p>This block of data allows the saving of all values wired to it in an external file. That file can be opened with the Notepad, and just by simply copying all values to the excel sheet we can get access to the graphic.</p>

These are some of the several blocks that are repeatedly used in the several programs ahead, because we have to have in mind that in order to the PID control function we have to constantly read the water level and send an output to the pump, only that way we can maintain a stable level of water. Some of the blocks above need more specification, like connection diagram and voltage limits, that information is presented on the Appendix A, in order to get a proper organization.

The next figure we have the Step Response of the system, considering the Upper Tank.

It's possible to see that the answer that it's showed isn't a perfect S shape, that could be because the amplitude of the step impulse and could also be due to the nonlinear response of the pump.

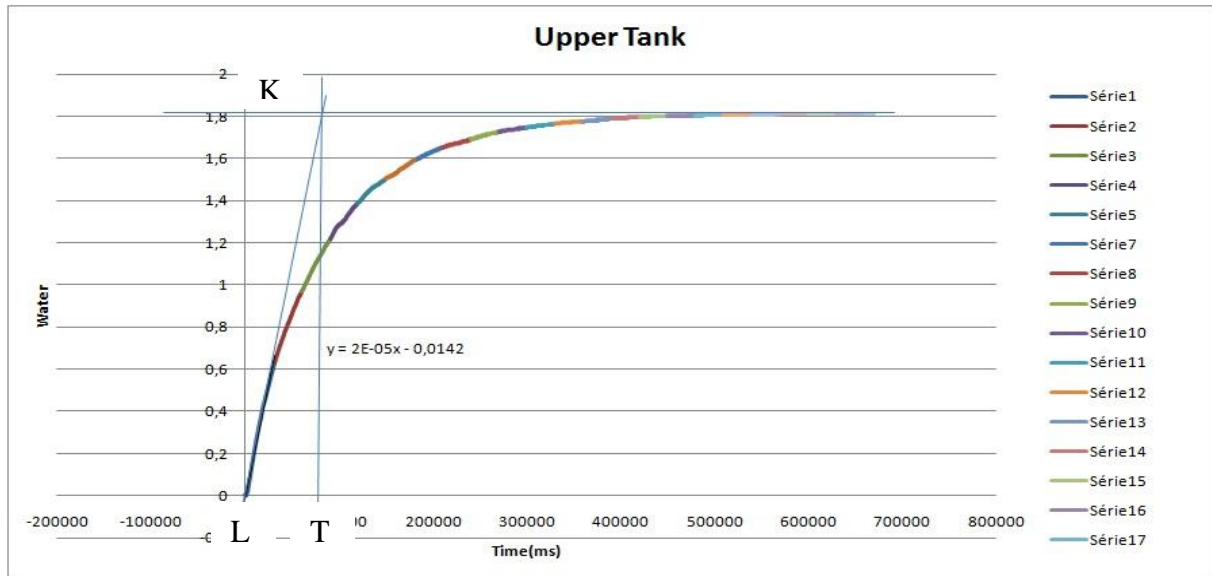


Fig. 11 - Upper Tank Tuning

The x-axis is in millisecond, but in every calculation it will appear in seconds, because it is the International System. For the Upper Tank we've got these values:

$$N = 2 * 10^{-5}; K = 1,85 \text{ cm}; L = 3,4s; N = \frac{K}{T} \Leftrightarrow T = \frac{1,85}{2E-5} = 92,5s$$

$$K_p = 1,2 \frac{T}{L} = 1,2 \frac{92,5}{3,4} = 32,7$$

$$T_i = 2 * 3,4 = 6,8$$

$$T_d = 0,5 * 34 = 1,7$$

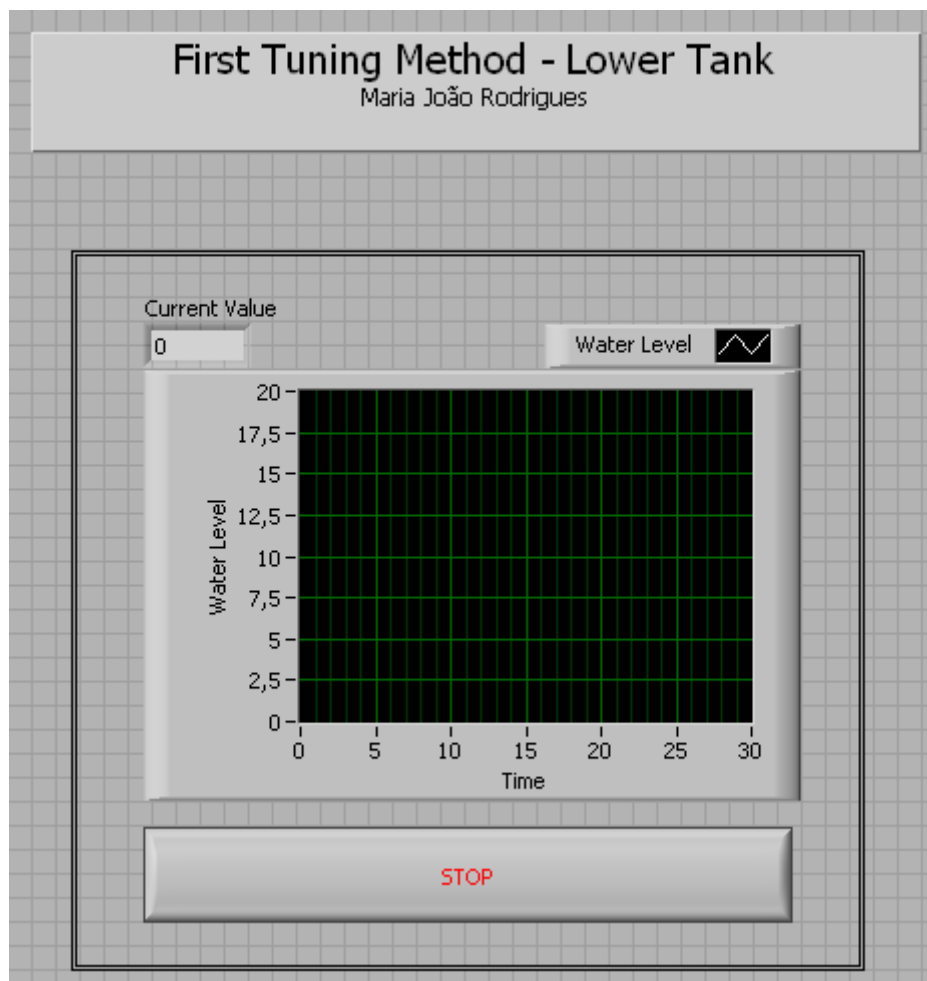
$$K_p=32,4$$

$$T_i=6,8$$

$$T_d=1,7$$

### 3.2.2 Lower Tank

The next image illustrates the work environment of LabView Upper Tank Tuning program. It is relative to the first method of tuning. It's pretty simple and allows the user to run and STOP the program whenever he wants; he also got access to the current value correspondent to the water level. The environment of the tuning program for the lower tank is very similar to the upper tank because both methods are the same; the only change is the tank that is working.



*Fig. 12 - Work Environment Lower Tank – 1st Method*

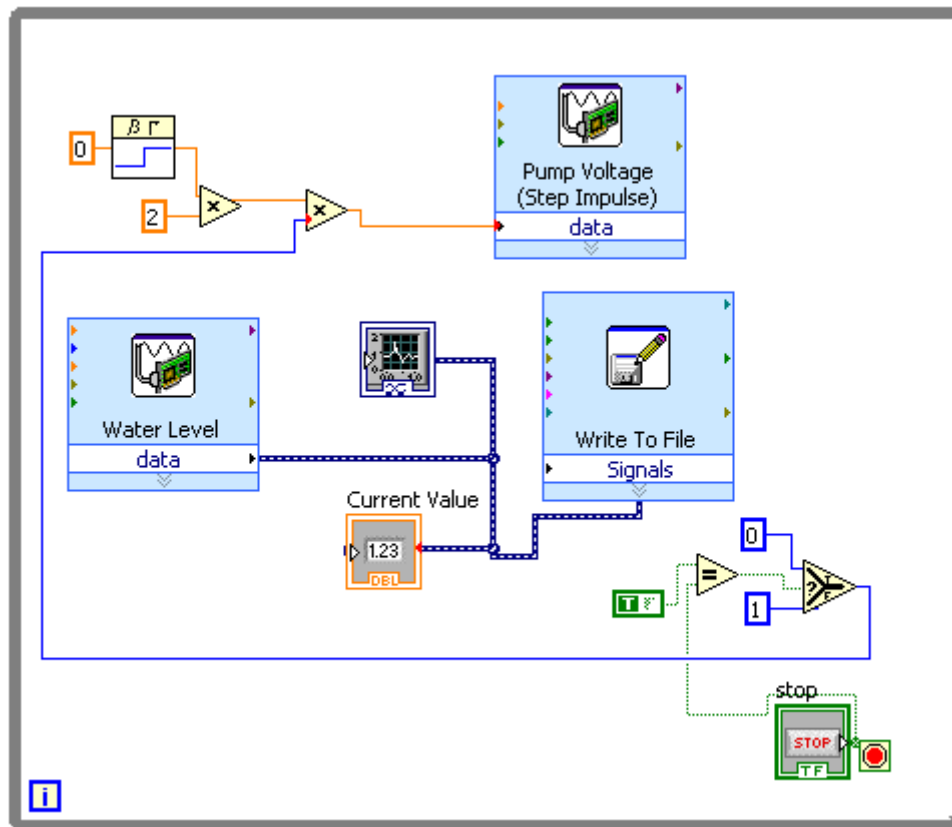


Fig. 13 - Block Diagram of the Lower Tank Tuning – 1st Method

The tuning of the lower tank is equal to the upper tank. The LabView program is very similar the only difference is that the file in which all the values are saved is different of course. The values of the lower tank are saved in a different file then the values of the upper tank.

This block do exactly the same that the ones before, the pump is getting an step impulse with an amplitude of 2 and at the same time the level of water is being written in a file, that way the step response (next figure) can be better analyzed.

All the explained blocks above, are the same for the lower tank tuning program too.



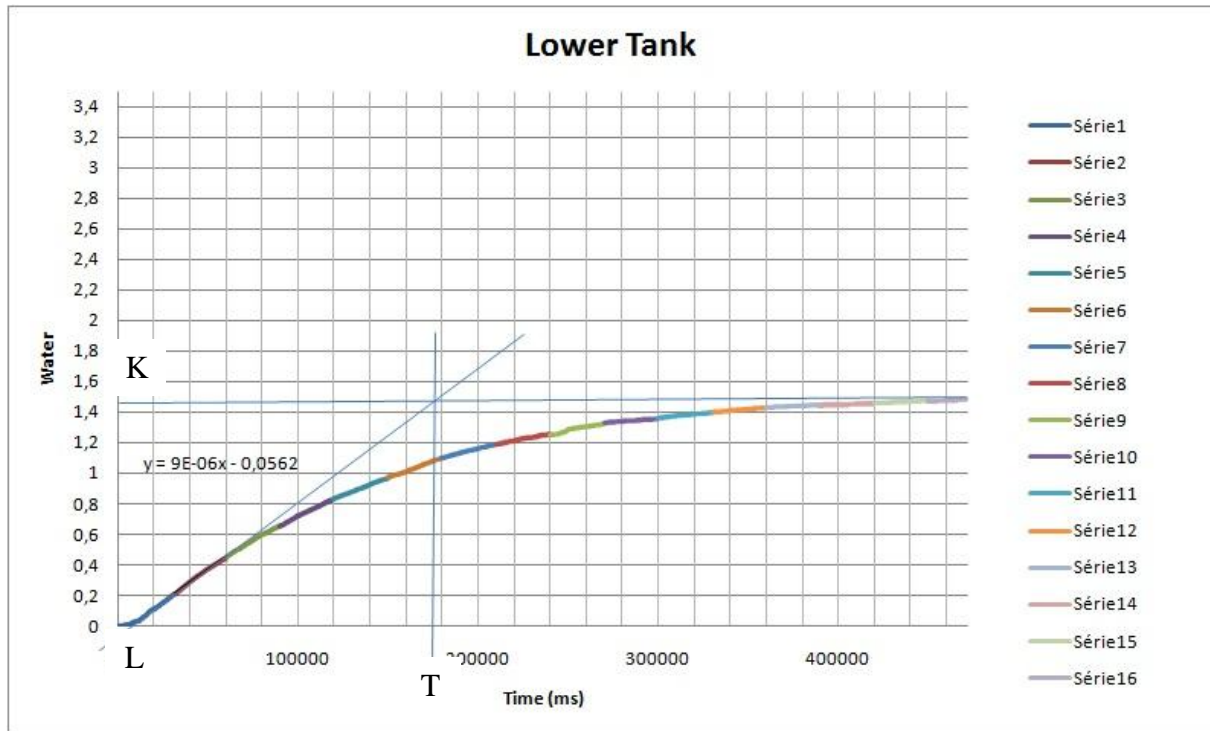


Fig. 14 - Lower Tank Tuning

The x-axis is in millisecond but every calculation was made in second because it is the International System. For the lower tank we've got these values:  $N = 2 \times 10^{-5}$ ;  $K = 1,49\text{cm}$ ;  $L=5,04\text{s}$ ;  $T = \frac{K}{N} = \frac{1,49}{2E-5} = 74,5\text{s}$

$$K_p = 1,2 \frac{74,5}{5,04} = 17,7$$

$$K_p=17,7$$

$$T_i = 2 * 5,04 = 10,07$$

$$T_i=10,07$$

$$T_d=2,52$$

$$T_d = 0,5 * 5,04 = 2,52$$



### 3.3 Second tuning Method

This method is only for the lower tank, because it's a method that implies that the system enters in state of disturbance and that disturbance couldn't be accomplished in the upper tank. In this method as said before, the P value is going to be increased until a disturbance is watched on the system; this system showed an increase and decrease on the level of water, that difference was always the same, that's when we know we can calculate the constants needed.

The next figure illustrates the work environment of the lower tank tuning program.

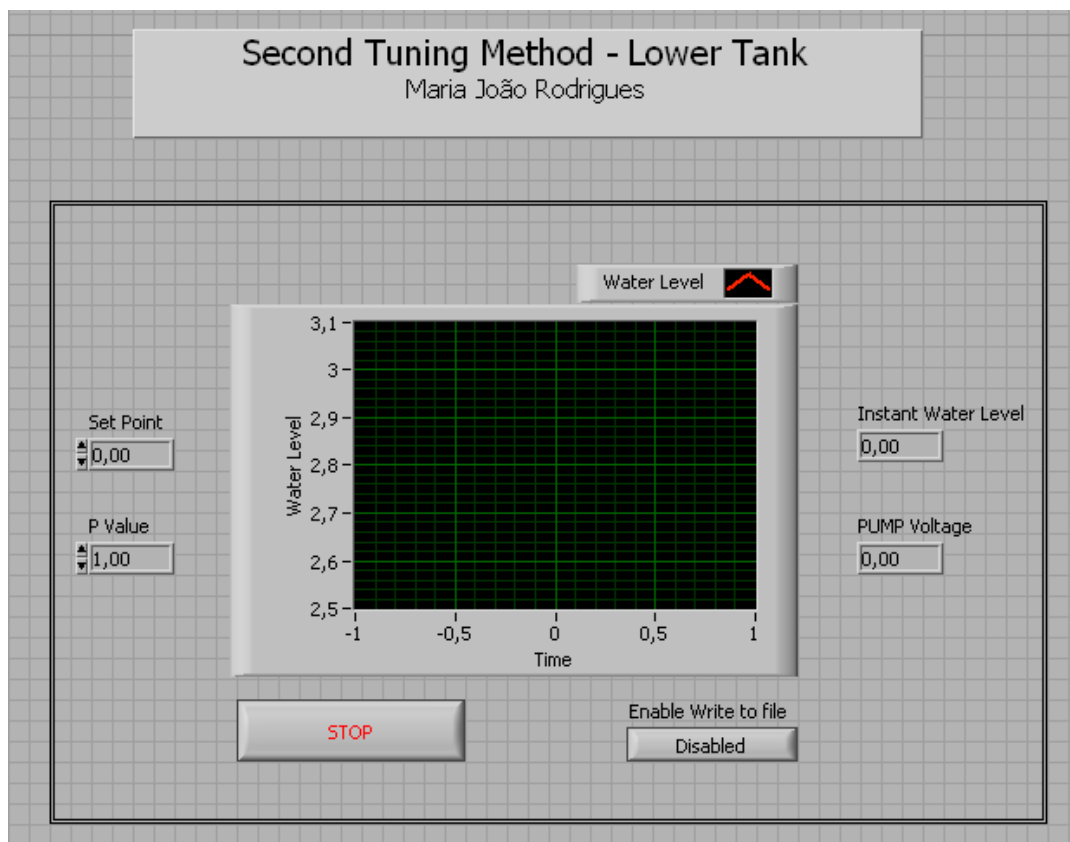


Fig. 15 - Work Environment Lower Tank - 2nd Method

In this environment we only needed the graphic to visually control the system status, the instant water level to have a more precision control. The Pump Voltage is present to, so that we can see the effect of increasing the P value on the Pump voltage. The set Point is where the user chooses the water level.

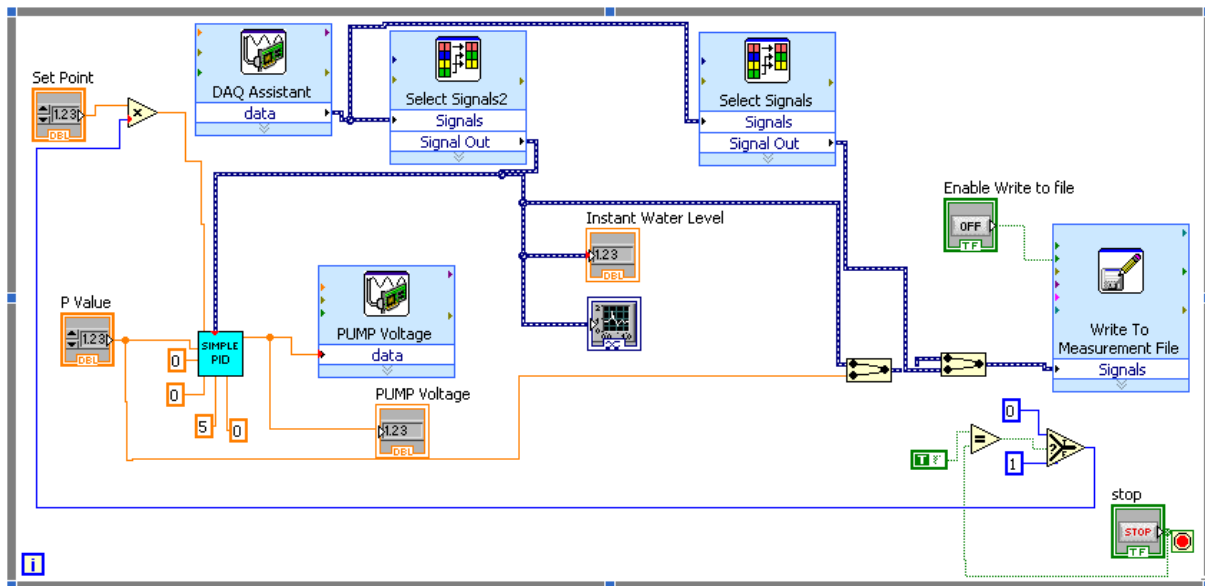


Fig. 16 - Block Diagram of the Lower Tank Tuning – 2nd Method

The Block diagram above contain mostly the same blocks above, the only difference is the Select Signals Block, which is explained forward.

Here, we have the PID block already present, because this tuning method is characterized by eliminating the Integrate and Derivate influence of the system and slowly increasing the Proportional influence in order to see a disturbance on the system. As we can see there are being saved three different variables to a file, the upper tank level of water, the lower tank level of water and the  $P$  value.

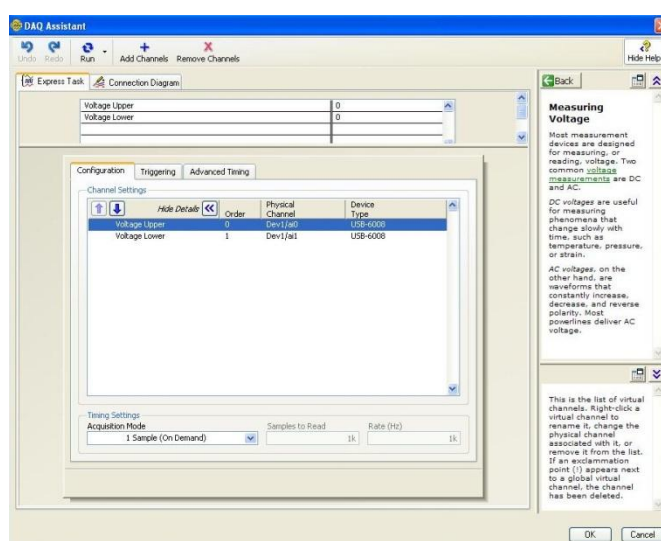
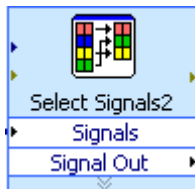
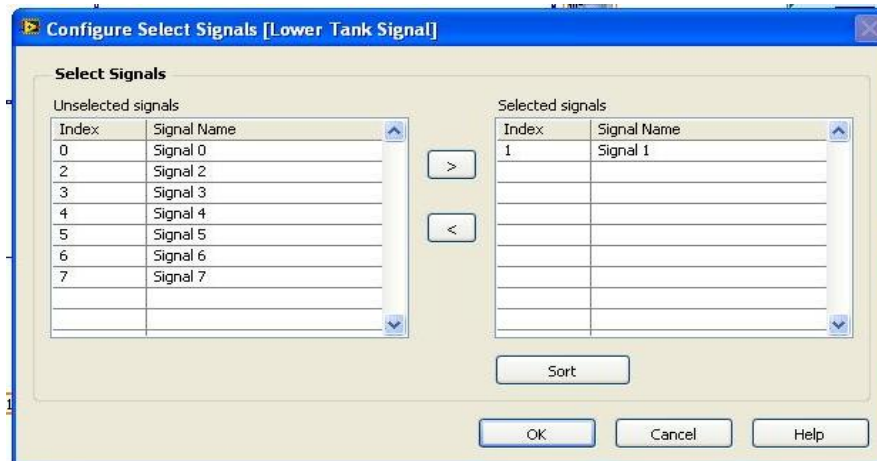


Fig. 17 - Connection Diagram - 2nd Method

Has it's possible to see, there are two channels being used in the same program. When this happens, we get the need to separate them into different paths. And that it's possible with the next block showed.



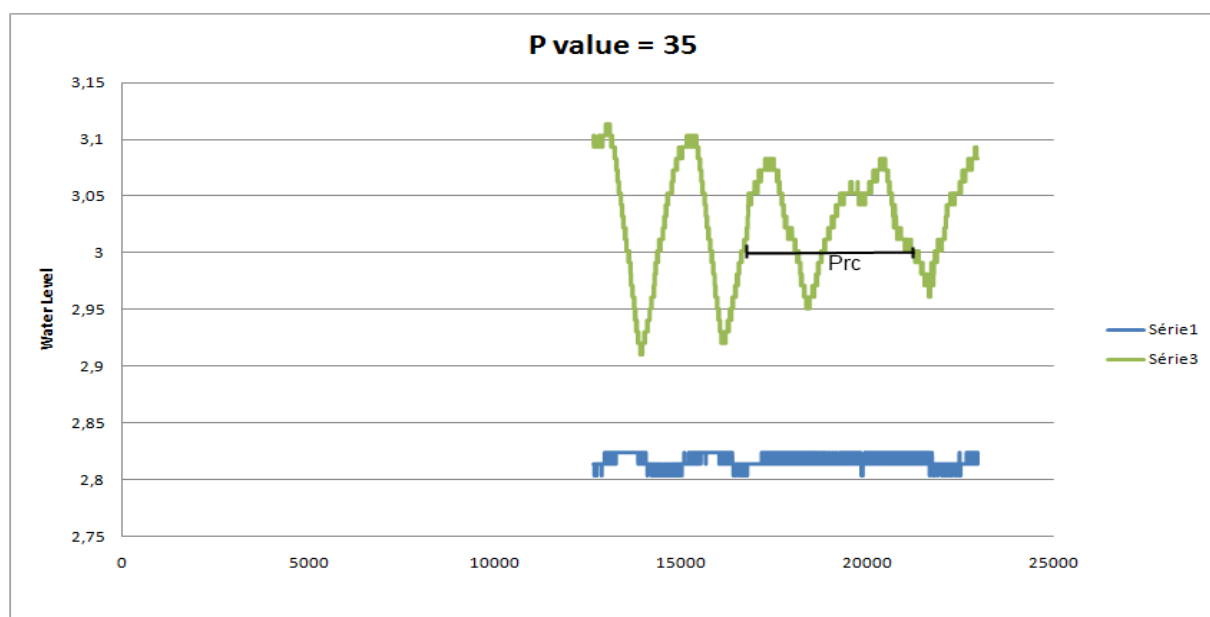
This block has the figure number 28 to complement it, this makes possible to in the same program acquire two different signals and separate them to two different paths.



*Fig. 18 - Select Signals Configuration*

As we can see this blocks shows all the channels available and the user only choose the one he would want to read values.

Next is shown the acquired values in form of a graphic, and as it's explained in the theory part, the period of the wave is very important. This wave illustrates the ups and downs of the water level; those ups and downs are the disturbance that we were looking for.



*Fig. 19 - Graphic for the lower tank - 2nd Method*

The figure number 29 shows the disturbance in the upper tank and the value  $P_{rc}$  it's the period of the wave, that give us information to calculate the  $K_p$ ,  $T_i$  and  $T_d$  values.  $P_{rc} = 44,00s$

$$K_p = 0,6 * P_{value} = 0,6 * 35 = 21$$

$$T_i = \frac{P_{rc}}{2,0} = \frac{44,00}{2,0} = 22$$

$$T_d = \frac{P_{rc}}{8} = \frac{44,00}{8,0} = 5,5$$

$$\mathbf{K_p= 21}$$

$$\mathbf{T_i=7,195}$$

$$\mathbf{T_d=1,79}$$

### 3.4 1st Method Vs 2<sup>nd</sup> Method

It's possible to see that the first and the second method result in different constant values, for that it's also possible to use both of the values that were obtained and compare them in order to see which method created the most accurate values.

For that we simply use both values, and monitored the response of the system to them. By simply compare them we can see which of them had a better response on the system, and by that is possible to affirm which tuning method is better, for our case.

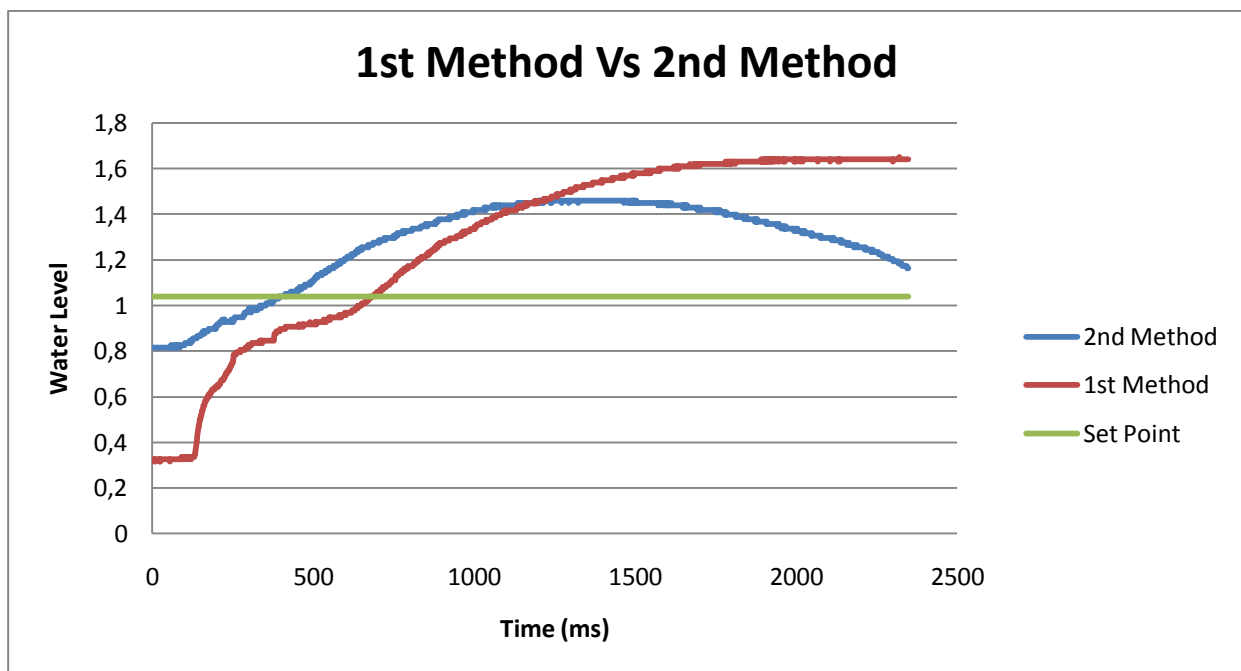


Fig. 20 - 1st Method Vs 2nd Method

Looking at the figure number 22 it's possible to see that the first method shown an overshoot by the SetPoint, that it's talked in the theory part, it could be happening because , as we can see the value of P is lower comparing to the second method. And it's known that to avoid an unstable system the programmer can lower the P value but that can lead to an overshoot like the one we have here. But a higher P value can also lead to an unstable system.

Table 2 - Tuning Methods - Constant Values

	1 <sup>st</sup> Tuning Method	2 <sup>nd</sup> Tuning Method
<b>K<sub>p</sub></b>	17,7	21
<b>T<sub>i</sub></b>	10,07	7,195
<b>T<sub>d</sub></b>	2,52	1,79

Table 3 - Constant Influence on the system

	To Big	To Small
<b>K<sub>p</sub></b>	The system may start oscillating.	The system can have less sensitive and don't respond.
<b>T<sub>i</sub></b>	It responds to pass values so it can overshoot the SetPoint. Eliminates the residual steady-state error.	
<b>T<sub>d</sub></b>	If the system is exposed to high levels of interferences the system can become unstable.	It slows the rate of change of the controller output.

As it's possible to see there are no perfect values for this kind of controls, the values are dependent of the system and also of the user choices.

The user can choose what kind of response he wants to his system and by that choose the constant values.

Here we are comparing two different kinds of values; the red ones were calculated through a program in open loop control, and the blue ones were calculated through a closed loop program control, it's important that the user knows it.

But the both of them were compared with in a closed loop control; the programmer just inserted them into the PID values of the PID control, and saved the answer of the system in order to be possible to compare them.

### 3.5 Main LabView Control Program

The next figure illustrates the work environment of the main control program in LabView.

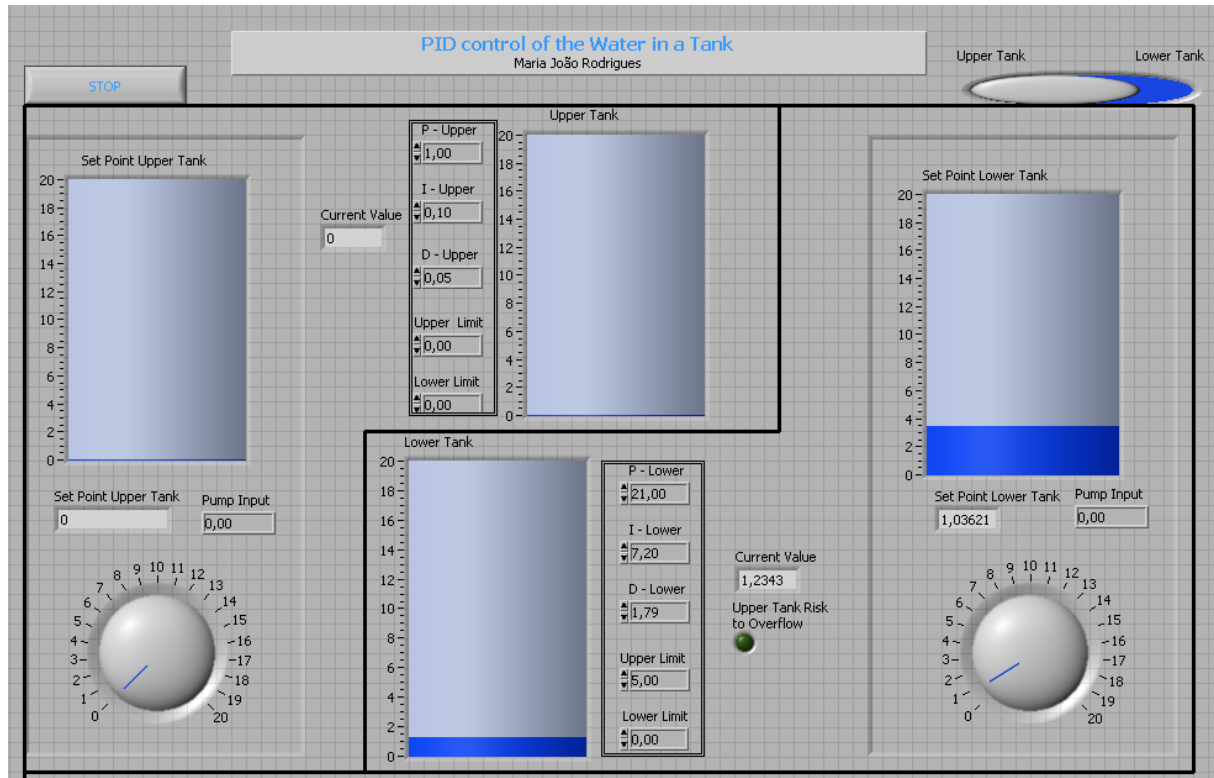


Fig. 21 - Work Environment - Main Control Program

In this work environment, the point was to keep it simple and useful. As we wanted to control only one tank at a time, the button on the right upper side of the screen enables the tank that the user wants to work with. If the user drags it to the left he will work with the upper tank, if the user drags it to the right he will work with the lower tank.

The upper tank system is really simple; it has a button in which the user can specify the water level that he wants and the right above tank shows that high in a more practical point of view. The current value is showed in the upper tank in the middle of the screen, right next to him is the PID values needed to work with the upper tank system.

The Lower tank system is also simple; the only difference to the upper tank system is that in this one we need to have a warning to the risk of the upper tank overflow. That's what that light is doing there, the light lights up if the upper tank shows any risk of overflow.

The next image is the block diagram of the main control program, for the upper tank.

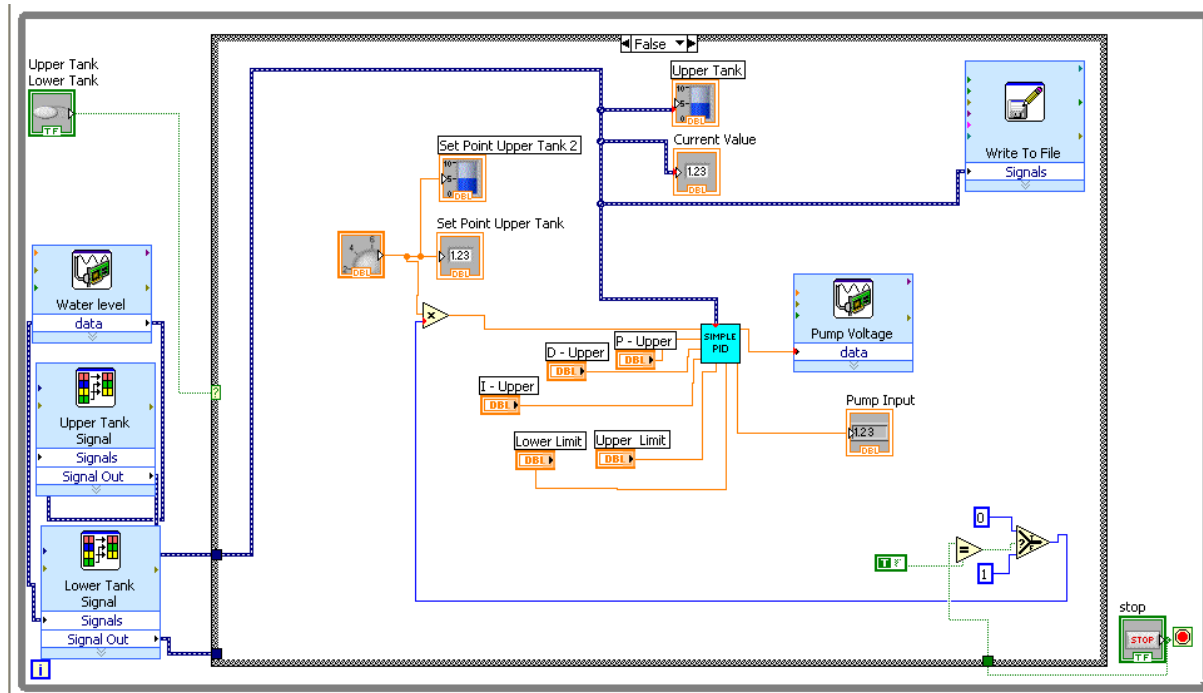




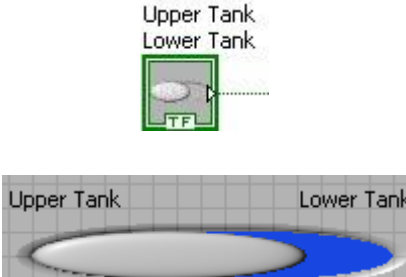
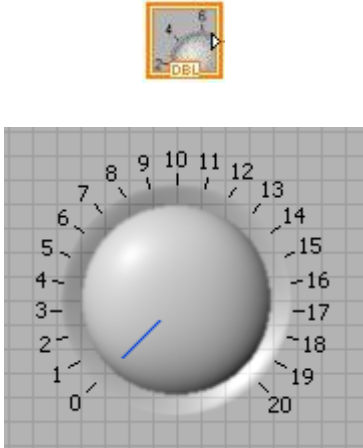
Fig. 22 - Main control program - upper tank

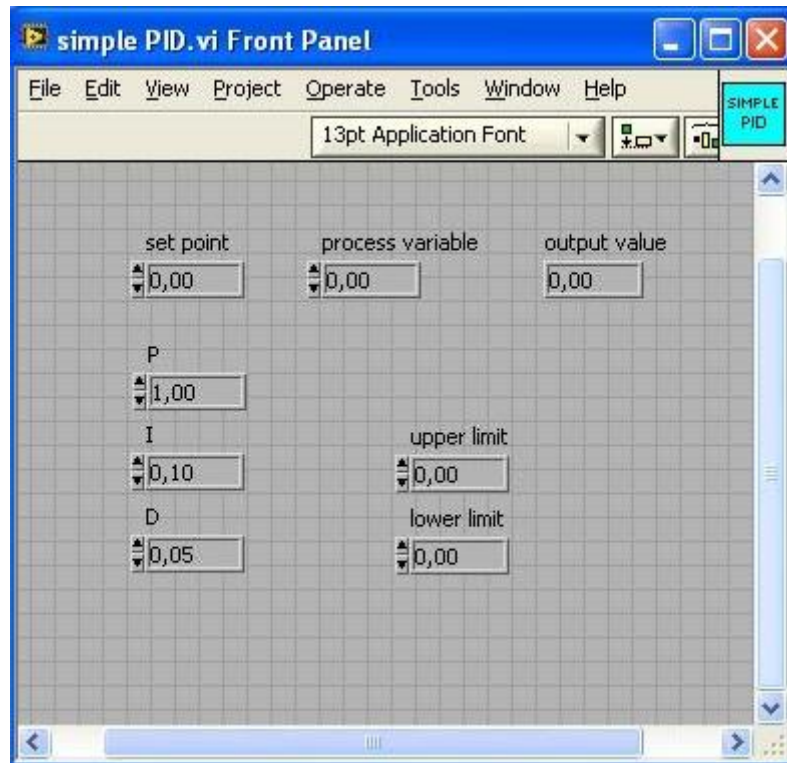
In order to be able to control each tank at a time, we need to create a case control; a case control verifies if a control variable, chosen by the user, is on a certain value, that value correspond to a case situation, that situation is chosen once again by the user, and depending on it, it depends the answer of the system.

So in our system, the button wired to the interior square, “Upper Tank Lower Tank”, controls the case situation, if it’s true the lower tank system starts working, if it’s false the upper tank work instead. This is a simple way of enabling and disabling certain parts of program blocks of data.

The next table there will be explained each and one of the blocks of data, how they work and what do they do in this specific program. It should never be forgotten that both systems, upper tanks system and lower tank system, are in a loop control, in order to be able to repeat all of these functions until the user feels like stop them.



	<p>This block has been showed before, is capable of reading input coming from outside. This means that the sensor, that measures the water level, is connected to the board. This specific block is acquiring sensors information, the sensor for the upper tank and the sensor for the lower tank. In order to separate information we use the next block.</p>
	<p>This blocks, are able to separate signals, with block named before we can read two signals, and then connect the <i>Data</i> output to the <i>Signals</i> input of this block. By doing that we are presented with a window that gives us the possibility to choose which sign we want to use. The <i>Signal Out</i> connection give us the sign that we chosen before. This block was used because the LabView didn't allow having two acquisition blocks, two blocks like the first one showed.</p>
	<p>This button is used to control the case structure, if it's true it will activate the lower tank system, if it is false it will activate the upper tank system, and of course, deactivate the other system. The upper image is the block image, and the lower is the environment image.</p>
	<p>This is a button that control the water level the user wants, the user just need to turn it around to the wanted value.</p> <p>The upper figure is the block image, and the lower figure is the environment image.</p>



The figure above it's the PID block, to this block is wired the P, I and D value, the Upper and Lower Limit, the SetPoint value, the Process Variable (the current value of water level), and the Output value. This PID block does all the calculations needed so there is only the need to wire all the values that he need to.

It is a predefined block of data, that the LabView program has available for its users, so therefore there is no need to create one from the scratch, knowing for sure that these one functions perfectly.

The next image is the block diagram to the lower tank.

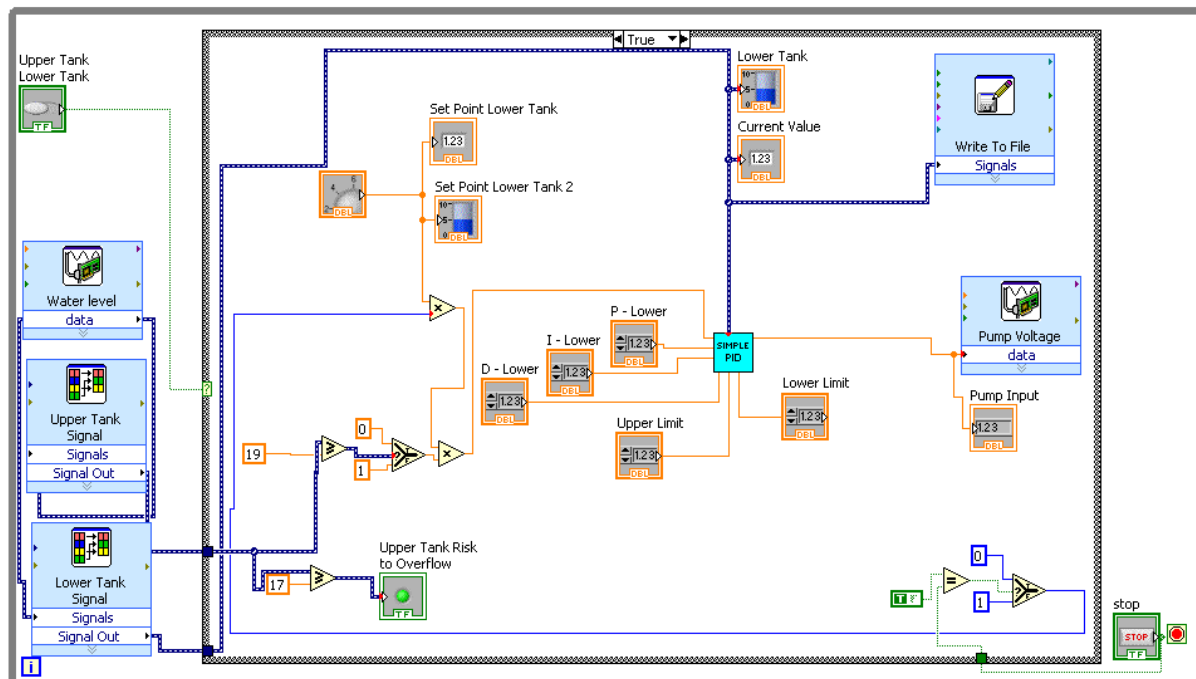


Fig. 23 - Main control program - lower tank

This control is equal to the previews one, the upper tank system, the only change is in the overflow controls, because when the lower tank is working it's important to verify if the upper tank is overflowing, and for that it was included certain specific blocks, explained next.

<p>The diagram shows a comparison block (greater than or equal to) with the value 19. The output of this block is multiplied by either 0 or 1, depending on whether the condition is met. This result is then multiplied by the Set Point to determine the final pump output.</p>	<p>This set of data it's called the overflow control. It will compare the water level of the upper tank with a defined number, in this case 19, and if it's equal or greater it will multiply the SetPoint by 0, and by that the pump will stop working. If it is not equal or greater to 19, it will multiply by 1 and nothing changes.</p>
<p>The diagram shows a comparison block (greater than or equal to) with the value 17. The output of this block is connected to a light indicator block labeled 'Upper Tank Risk to Overflow'.</p>	<p>This system is verifying the level of the water of the upper tank, and if it's equal or greater than 17 it will light up the light, in order to warn the user about the risk of overflow.</p>

## 4 Discussion

For the upper tank the values are really good, they do not are very different from the usually values, for the same system. The answer illustrated on the graphic, should be a perfect S shape, but that wasn't possible due to non linear response of the pump. Anyway, we were able to calculate the L value the same and get a conclusion.

On the lower tank we were able to consider two tuning methods, and by that we could perfectly compare the values obtained in the two. The first method it's the same for the upper tank, all the values were obtained through an open loop control program, and they were like was expected.

In the second method all values were obtained in a closed loop control system, using already the PID controller, but setting the derivative and integrative influence to zero, and then by increasing the proportional term we were able to see when the system enter the oscillation and calculate the two other constants. They were different from the first tuning method but that don't necessarily means that something is wrong. Just means there are two different results that could work in this system, and depending on the answer we want.

And actually they aren't so different from each other, they are quite similar but even then they have quite different answers on the system, the first method cause an overshoot by the SetPoint, that it's expected because a lower value for the proportional influence can cause that kind of answer on the system. That «big» overshoot that is presented for the first tuning method doesn't exist for the second tuning method and it's due to the higher value of the proportional influence, but that doesn't mean that the second tuning values are better than the first ones, because a high proportional value can lead to an unstable system, and sometimes there are system that cannot become unstable.

About the program that was used, LabView is simple to use and every program that was built had the point of being simple to use by the user and avoid being too complicated to see and comprehend. And with LabView that can be quite easy, due to the several decorations and even the because of the buttons, tanks and other animated and interactive gadgets.

It's a very fast and powerful tool, and made the programming much easier. The possibility of using figures in the shape of a tank, and the animation of the tank increasing the water level and decreasing gave and extra something to the work. All the buttons were very useful and they will simplify the work of the user.

The board NI USB-6008, is also very simply to use, it's perfectly functional with the LabView, every connection and functionality of the board were active. The board has two outputs of 5V, and several inputs. All configuration associated to the board is easy to make and very fast.

## **5 Conclusions**

As we were able to see depending on the tuning method the results may change a bit, that doesn't mean that any of them are wrong, just different. And to different values we have different results, and by that we have to analyze which response is the better for the system that is in use.

It's easy to see that a PID control can be applied to many feedback systems, when the need to constantly control a system variable appear the PID control it's a good solution. Quite easy to use and create. It only requires the need of tuning, but even that isn't so hard to do. There are quite good information about it, so should not be hard to do it.

Talking about the LabView programming, it is also very simply to use. Besides the help that any programmer can get by just asking a LabView Professional, the actual program has a help menu that gives good and important information about every block of data, with that any programming get's quite easy to complete.

## References

- [1] Åström, K. J. e Hägglund, T. ,PID Controllers: Theory, Design and Tuning, 2nd, 1995,pp 59.
- [2] Gene F. Franklin, J. Davied Powell, Abbas Emami-Naeini, “Feedback Control of Dynamic Systems”, 6<sup>th</sup> edition, 2010, pp 19 – 27.
- [3] Adian O’Dwyer , “Handbook of a PI and PID Controller Tuning Rules”, 3<sup>rd</sup> edition, 2009.
- [4] National Instruments, DataSheet NI-USB 6008. (<http://sine.ni.com/ds/app/doc/p/id/ds-218/lang/en>)

## Appendix A

Appendix A is for all the extra images that do not are essential to the full comprehension of the report.

- Upper Tank LabView Program

The next figure, illustrates the channel chosen to the readings. The blue line shows that the input will be a Voltage, on the Physical Channel: Analog Input Number 0 with a maximum of 20V and minimum of 0V. In this configuration step it is possible to run the channel and see if it's working properly.

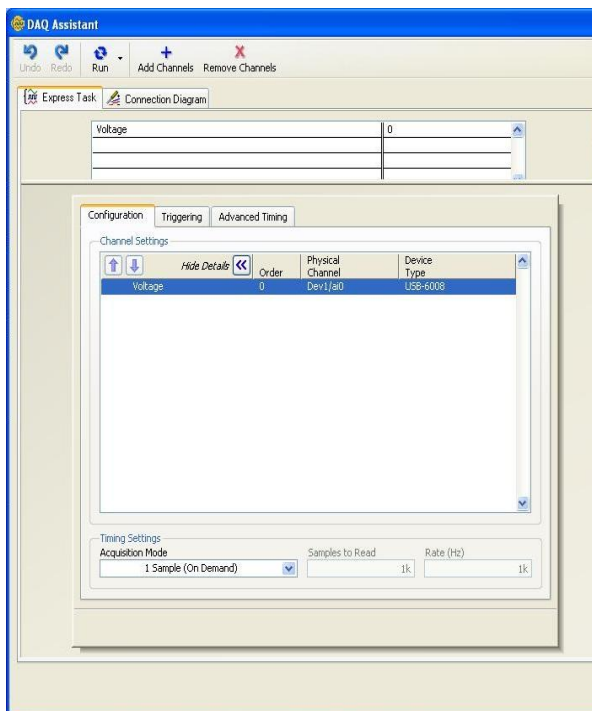


Fig. 24 - Express Task Upper Tank Tuning

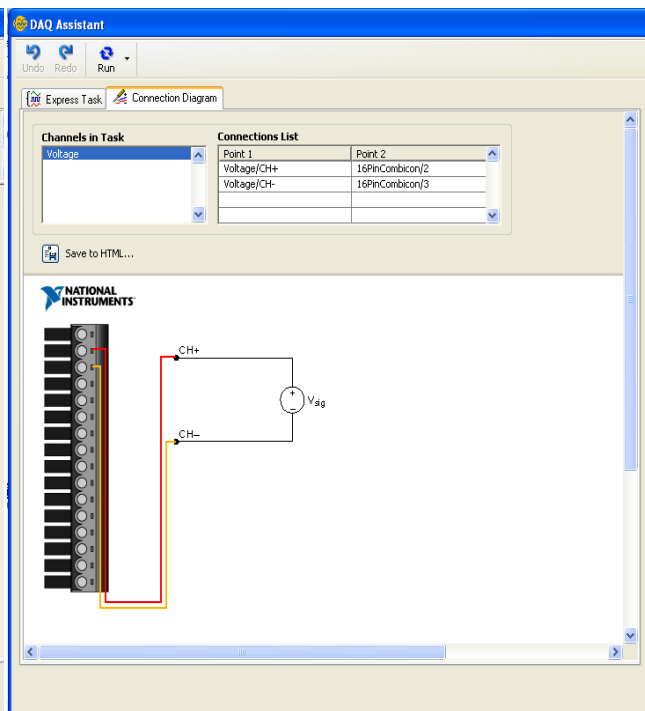


Fig. 25 - Connection Diagram Upper Tank Tuning

The Fig.23 clearly illustrates the connection to the board, there is being used the channel number 0, that means the connections 2 and 3 are used, positive and negative.



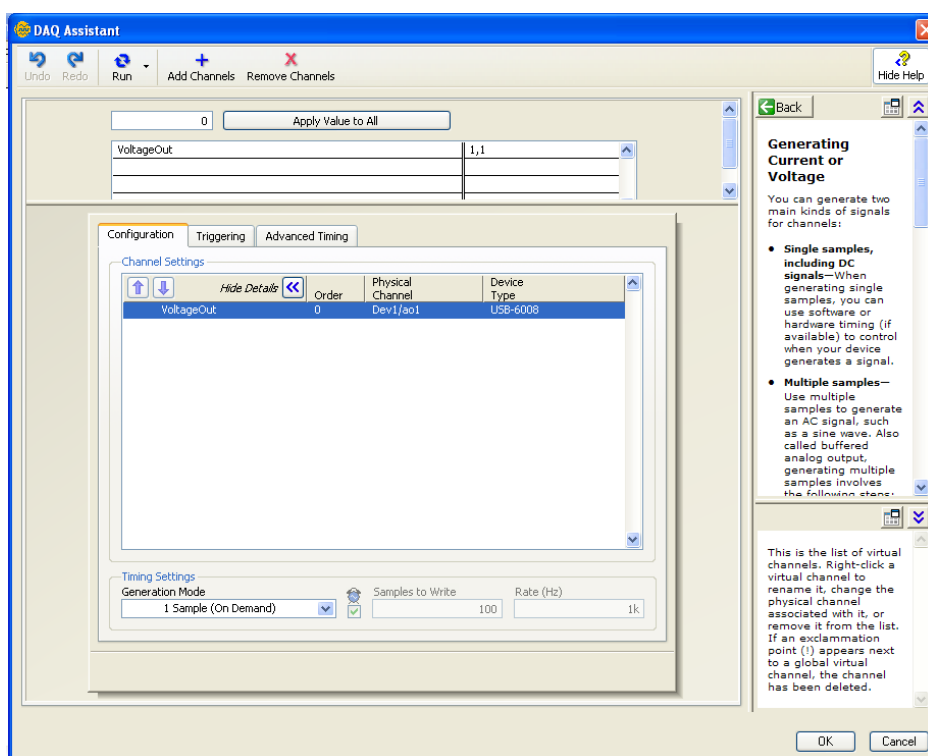


Fig. 26 - Configuration Pump Input – Upper Tank

This is the screen to the configuration of the output to the pump; it's possible to see that we got access to the channel of the Analog Output Number 1, which means that the connections 15 and 16 are used. It's very important that in the Generation Mode we choose: 1 Sample on demand, which is because we only want one sample when the user demands.

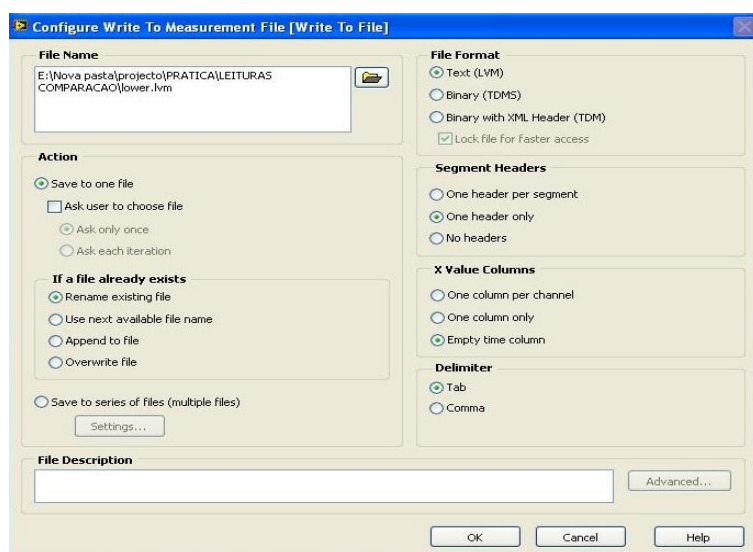


Fig. 27 - Write to File Configuration Form– Upper Tank

In this figure we the configuration of the save to file option. The File name option let us choose the path where to save the file. All of the other option can be chosen by the programmer, and mostly regarding appearance.

- Lower Tank LabView Program

All the preview images are the same to the lower tank has said before. The only change is in the name of the file that saves all the values.

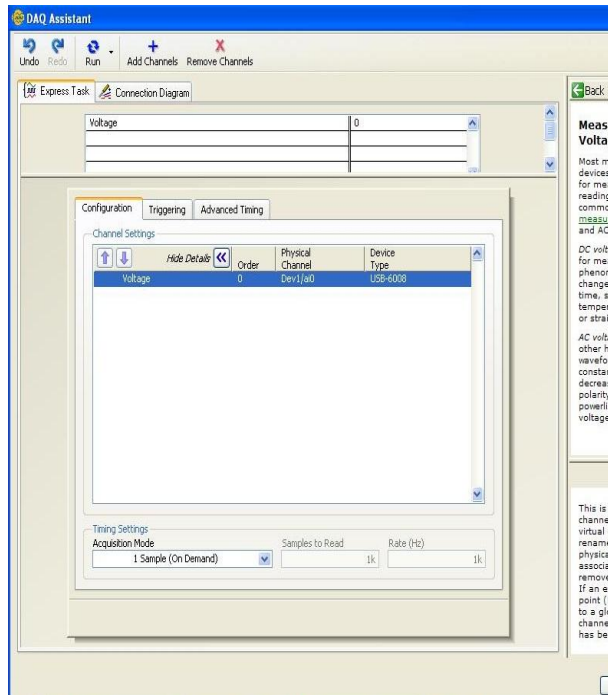


Fig. 28 - Express Task Lower Tank Tuning

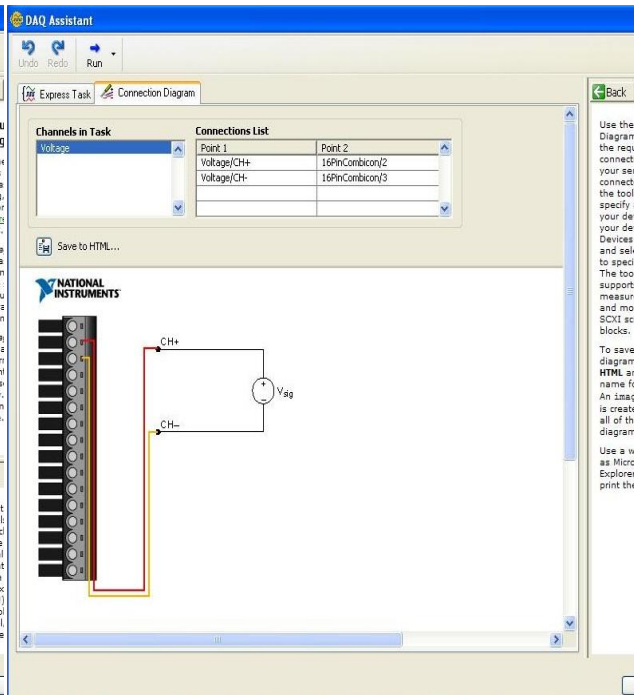


Fig. 29 - Connection Diagram Lower Tank Tuning

As we can see, even the connection diagram is equal to the connection diagram of the upper tank because we are talking about two different programs, if they were in the same program this couldn't be possible. Here again the maximum voltage is 20V and the minimum voltage is 0V.

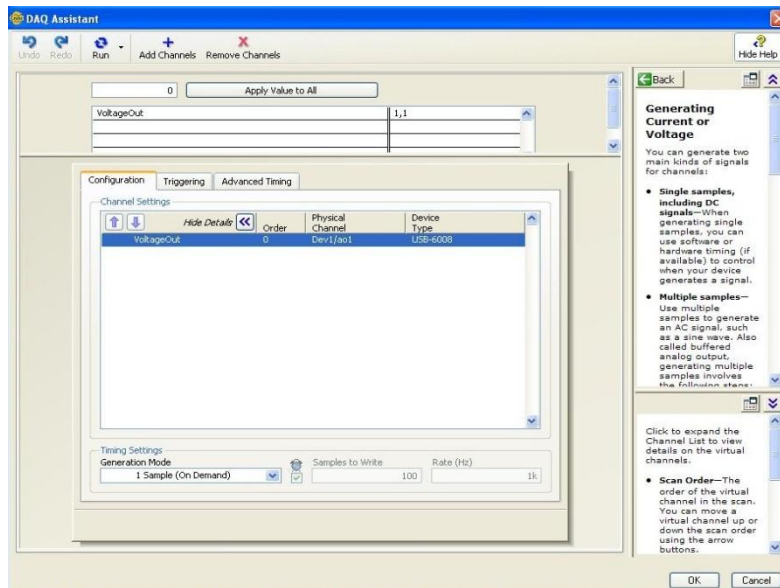


Fig. 39 - Configuration Pump Input – Lower Tank

The previous image is the configuration form to the pump Input; here we can see that the channel that was used was the number 1, which means that the inputs 15 and 16 were being used. The next image shows the configuration form of the Save to File block, it's equal to the upper tank form the only change is the path and the name of the file.

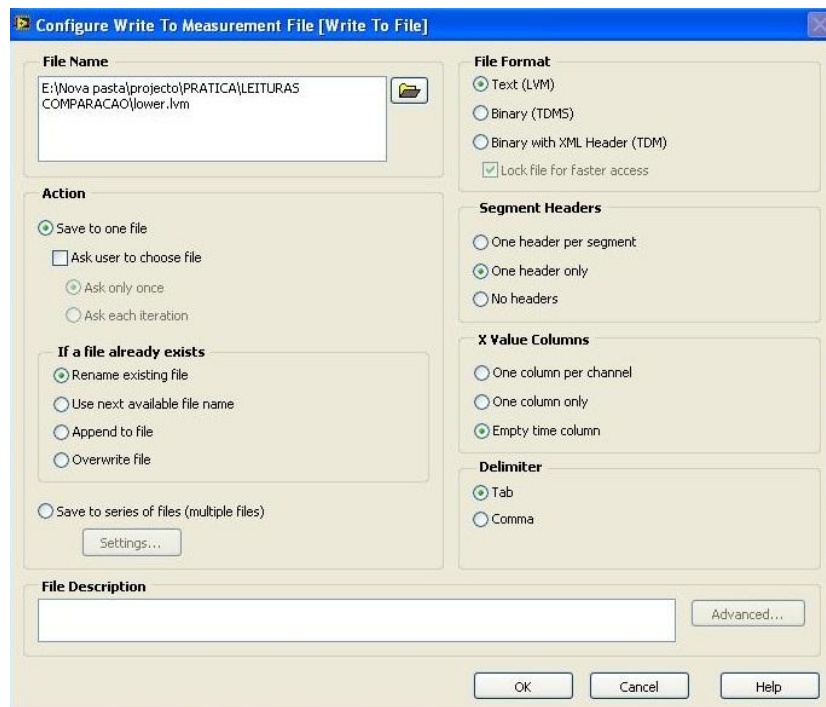
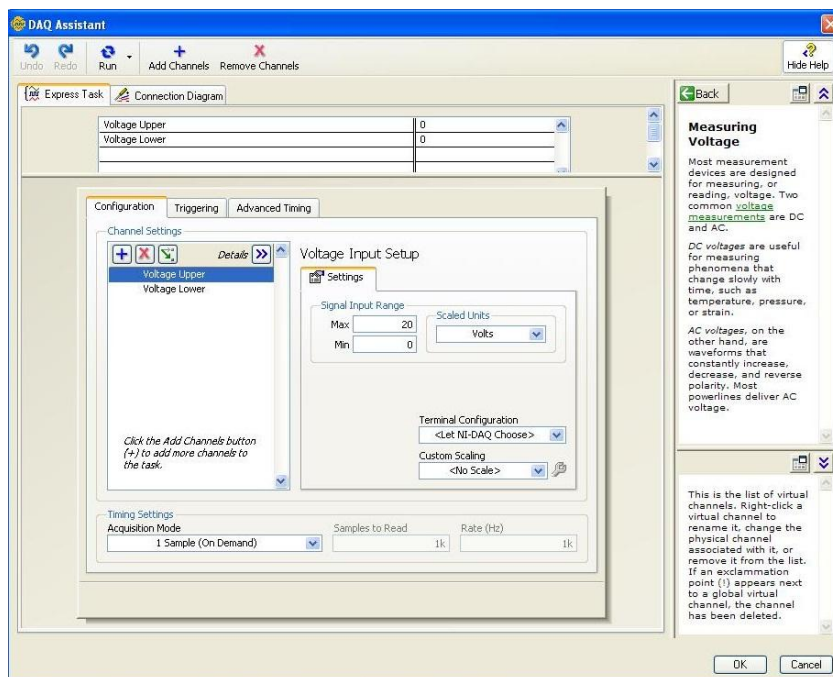


Fig. 40 - Write to File Configuration Form – Lower Tank

## Appendix B

Next it will be presented all of the non necessary figures; essentially they illustrate the connection needed in the board. These images are regarding the Main Control Program.



This image represents the both signals that are acquired on this program. Both have a maximum voltage of 20V and are in the Acquisition Mode of 1 Sample on demand.

Fig. 30 - Connection Configuration

The next images are the connection diagram. They illustrate the connection to the board.

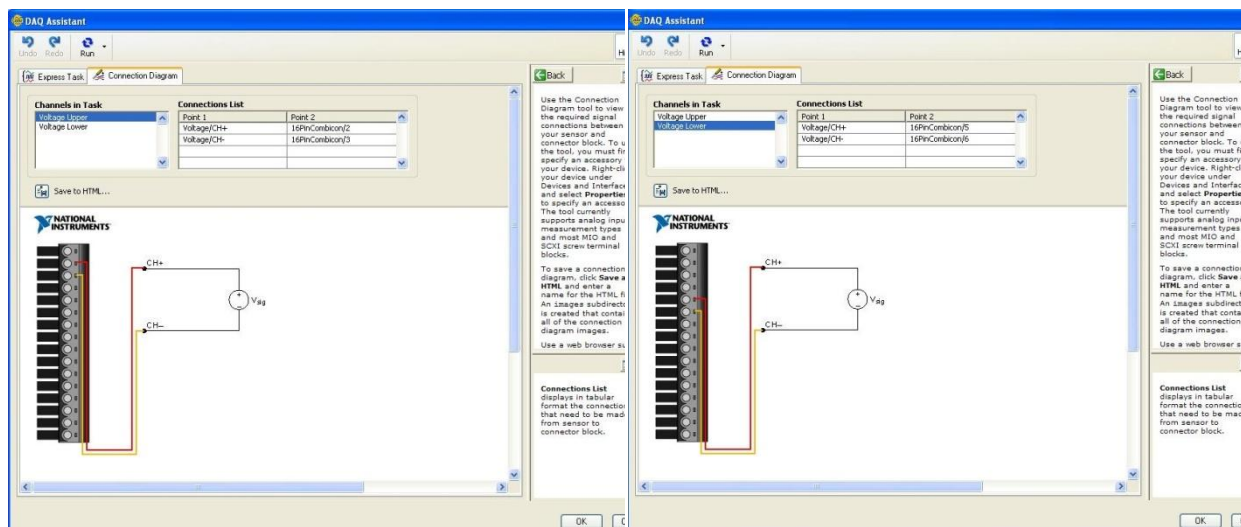
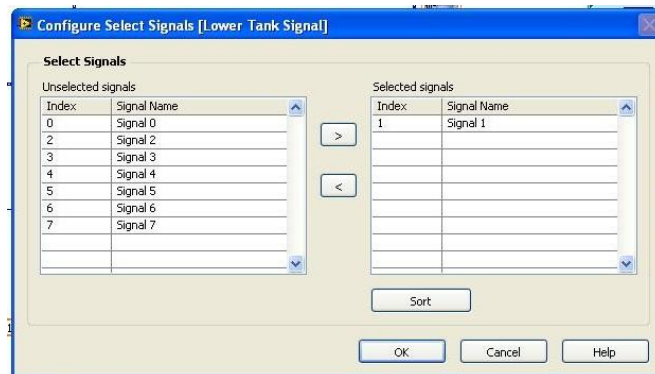


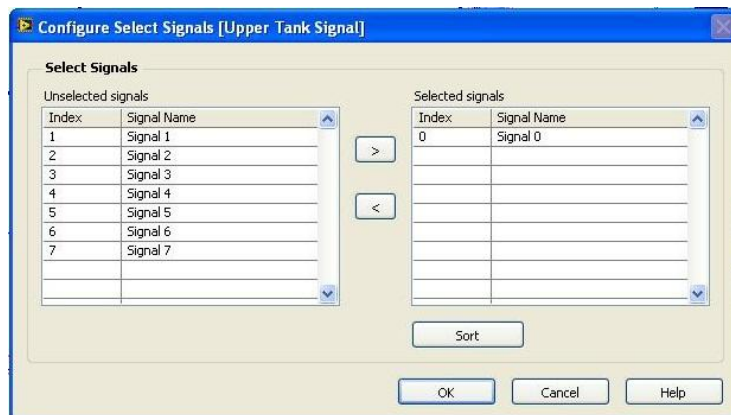
Fig. 31 - Connection Diagram - Upper tank

Fig. 32 - Connection diagram Lower tank

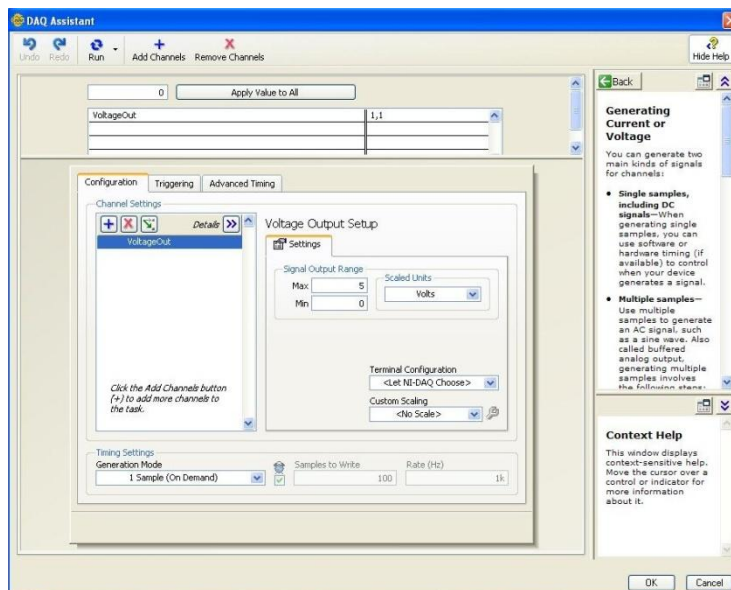
The next images represent the select signal talked before. We have on the left side all of the signals available, and on the right side we have the signal that we want to use.



*Fig. 33 - Select signal 1*



*Fig. 34 - Select signal 2*



This image illustrates the output configuration, here we can see that the maximum voltage is 5V, and once again the Generation Mode is on 1 Sample on demand.

Fig. 35 - OutPut Configuration

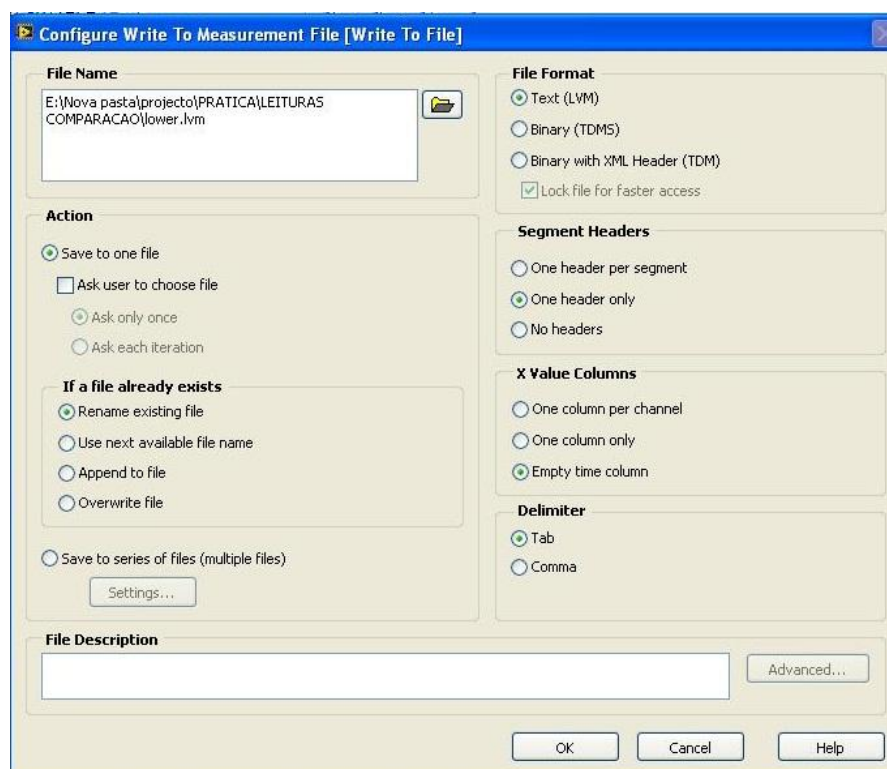


Fig. 36 - Save to file Configuration

The above image is the configuration window to save to file. This window allows us to change the path of the file and its name.