Journal of Software Engineering and Applications, **2011, 4, 161-171**
doi:10.4236/jsea.2011.43018 Published Online March 2011 (http://www.SciRP.org/journal/jsea)

A Design of a PID Self-Tuning Controller Using LabVIEW

Mohammad A. K. Alia, Tariq M. Younes, Shebel A. Alsabbah

Mechatroncis Engineering Department, Faculty of Engineering Technology, Al-Balqa Applied University, Amman, Jordan.
Email: makalalia2000@yahoo.com, tariqmog@hotmail.com, shebel_asad@hotmail.com

Received February 20th, 2011; revised March 5th, 2011; accepted March 9th, 2011.

ABSTRACT

In this paper a trial has been made to design a simple self-tuning LabVIEW-based PID controller. The controller uses an open –loop relay test, calculates the tuned parameters in an open loop mode of operation before it updates controller parameters and runs the process as a closed-loop system. The controller reacts on a persistent offset error value as a result of load disturbance or a set point change. Practical results show that such a controller may be recommended to control a variety of industrial processes. A GUI was developed to facilitate control-mode selection, the setting of controller parameters, and the display of control system variables. GUI makes it possible to put the controller in manual or self-tuning mode.

Keywords*: *PID Control, Manual Tuning, Self-Tuning, Open-Loop Relay Test, Process Variable, System Offset Error

1. Introduction

Proportional integral derivative (PID) control method (algorithm) has been the most popular control method, which is widely used in control engineering. From an automation perspective, PID is more than enough for 99% of control situations. It is well known that a great many systems have very simple dynamics and in these situations PID is often sufficient to provide the performance needed. Many industrial control loops that are nonlinear to some degree are linear enough in the control region near the set point for which PID control algorithm works fine. PID controllers can be more intuitive to tune. For example it is easier to reason out the expected behavior while changing one of the PID gains than pole placement option. There are many advantages of PID's such as their simplicity and possibility of coupling PID algorithm with smith predicator, feed-forward loops, nonlinear gain scheduling and other advanced control techniques [1,2].

PID controllers have some drawbacks that limit their effectiveness. One of the current difficulties with PID controllers is the gain tuning. Although there are autotuning algorithms available yet an experienced engineer is still required to fine tune the controller and ensure system stability. In the majority of cases PID tuning involves trial, (and) error and direct intervention of the operator during the tuning process particularly during disturbance tuning when it is not always obvious whether the process variable is reacting to the control effort or to additional disturbances or measurement noise [3-5]. Another drawback of PID controllers is that process dynamics might change over time. This can happen due to variation of (changes of) the process load, and normal wear and tear. To compensate for process behavior change over time, expert users are required to recalibrate the PID gains. This drives up costs for labor and down time. In order to eliminate the need for operator intervention, it is recommended that control tuning be enabled when the process variable begins a limit cycle [6,7], which may be detected easily, also it contains enough information for determining a new set of controller parameters. This is what we are going to do.

2. Basic Idea and Design Considerations

The target of this work is to design a LabVIEW–based self-tuning PID controller and to verify its performance using a process flow-rate trainer which exists at the laboratory of process control. For this purpose an ISA standard form of PID algorithm was designed. In order to eliminate the effect of external noise on measurement, a low pass fifth order filter (FIR) is used. For controller to react on the controlled system error only, and to avoid flattering, the controller initiates the tuning process only

after the existence of an error for a predetermined time interval. In order to realize the required time delay a specific ON- delay timer was designed in compliance with the concept of data flow programming [8]. The On-delay timer is illustrated in **Figure 1**.

The range of set point is (0-2) V while the controller output range is (0-10) VS. In order to avoid problems associated with closed-loop tuning [4-5], the controller applies an open-loop set point relay test by forcing the process variable into a series of sustained oscillations known as a limit cycle. The operator has the choice to select the appropriate number of oscillations. In order to obtain more tuning accuracy it is preferred to increase the number of oscillations and take the average (T_U). Once the parameter settings have been loaded into the PID formula, the controller is returned to the automatic mode. In this connection we should refer to the basic difference between the designed self-tuning controller and the auto-tuning controller toolkit of LabVIEW [9]. We do not use a wizard and human intervention is excluded more over we apply an automatic open-loop tuning procedure instead of closed-loop tuning.

Controls:

- K_p: the value of the proportional controller gain.
- K_i: the value of the integral gain.
- K_d: the value of the derivative gain K_d.
- SP: desired steady state value.
- Bias: the value added to controller output and when the error equals zero the output equal the bias value.
- SS Ripple band: the accepted value of ripple for process variable in order to consider that steady state occurred.
- Relay Amplitude: the value of Sp relay.
- # of Cycles: the number of cycles that must occur to stop tuning.
- Mode: a select switch between automatic and manual mode.
- Manual Control: the value to send at manual mode.
- Manual: makes automatic to manual modes change.
- Auto Tuning: starts auto-tuning operation.
- Stop: the abort push button.

Indicators:

- PV: the current value of process variable.
- Error: the value of subtracting current PV from SP.
- Out: the value of current controller output in volt.

4.2. The Block Diagram

The block diagram is shown in **Figure 4**. It includes the PID auto-tuning (all features) SubVI, Filtering and smoothing wave form SubVI, and the waveform values average SubVI.

The hierarchy of the VI is shown in **Figure 5**.

3. PID Self-Tuning VI Flowchart

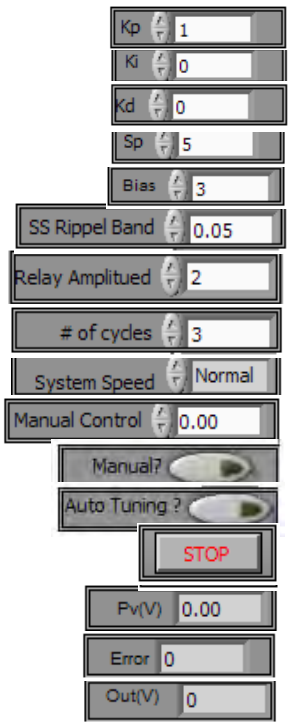
Figure 2 shows the flow chart. When the VI runs it reads the data from the analog input and filters the process variable. If the auto-tuning is activated by the auto-tuning pushbutton or as a result of disturbance, the controller switches to the open-loop mode.

When the system is in steady state the tuning process starts. When a limit cycle exists with the required number of cycles, new PID parameters are evaluated, then PID controller parameters are updated and the auto-tuning process is stopped. If the stop pushbutton is not pressed the controller goes to next iteration, else it sends a zero output and stops. When the auto-tuning is off, the controller works as a normal PID controller, then it adds the required bias to its value and coerces it in range if the stop pushbutton is not pressed.

4. Description of Self-Tuning PID Controller VI

4.1. The Front Panel

The front panel is shown in **Figure 3**. It includes the following controls and indicators.



The PID auto-tuning (all features) SubVI is shown in **Figure 6**. It includes the following SubVIs:

4.2.1. Restart Response VI

It sends zero volt as a controller output when it is enabled. When the process variable (PV) becomes zero the process starts again. The block diagram is given in **Figure 7**.

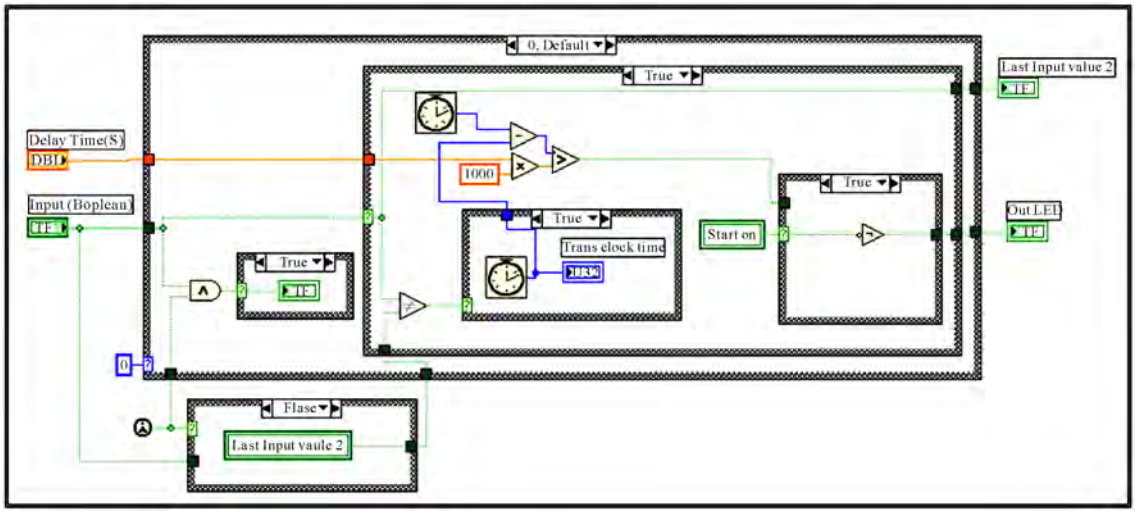


Figure 1. On-delay timer block diagram.

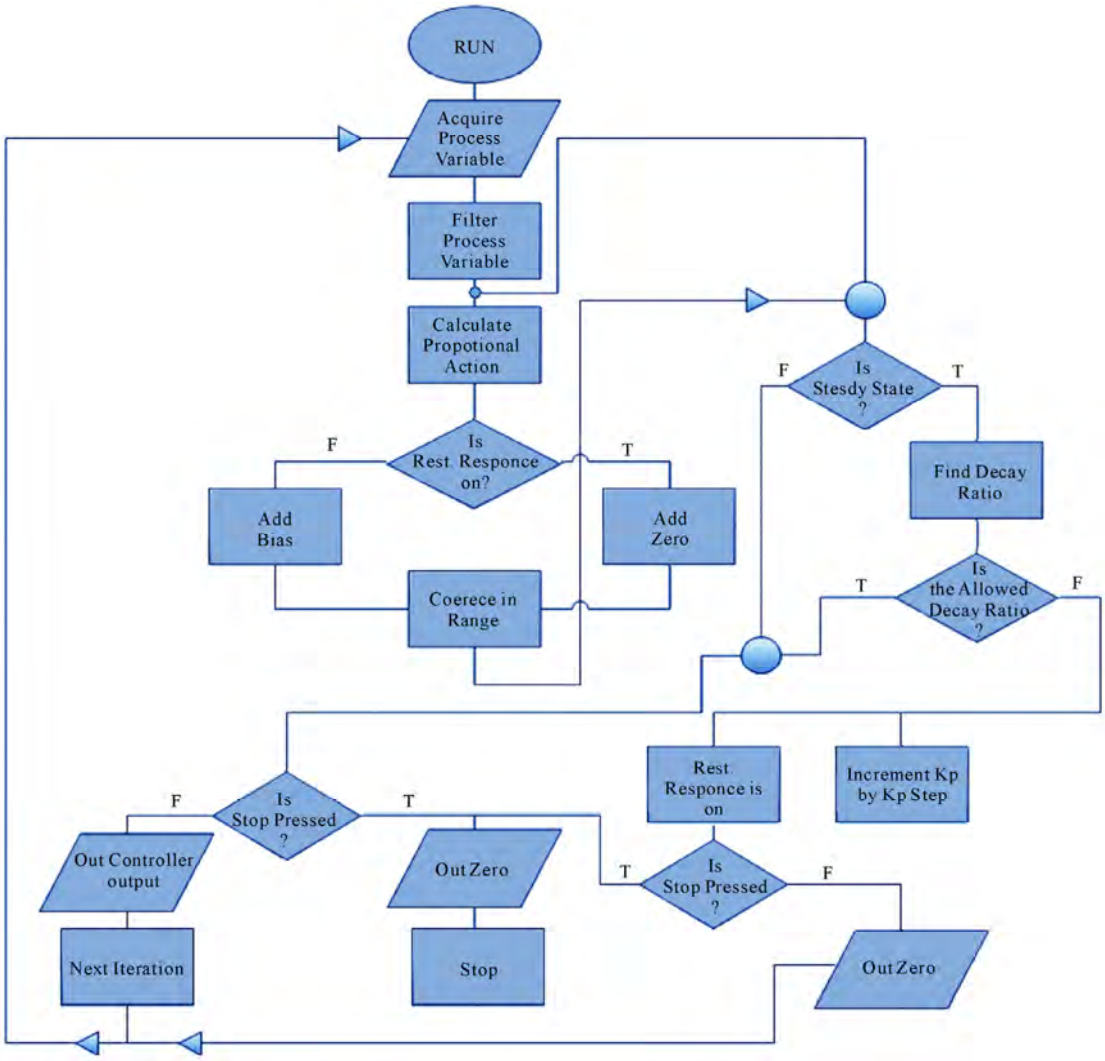


Figure 2. PID auto-tuning VI flow chart.

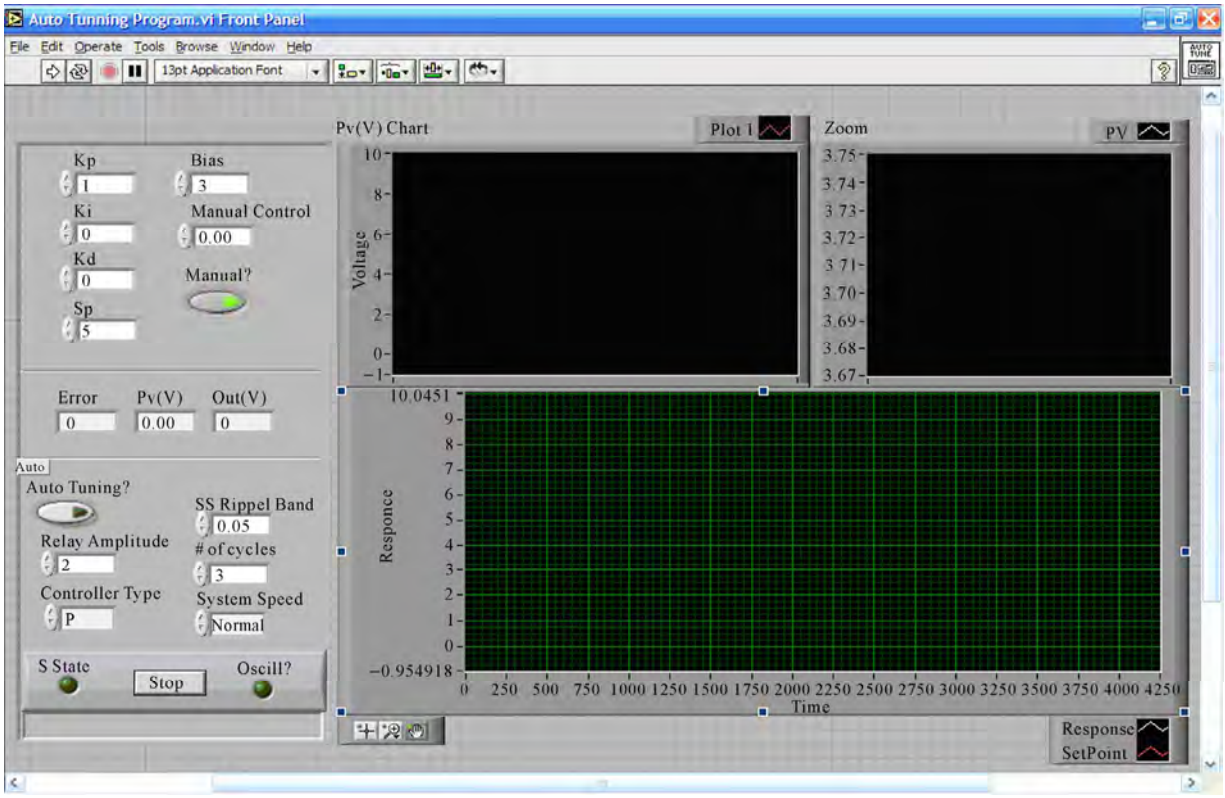


Figure 3. The front panel of PID auto-tuning VI.

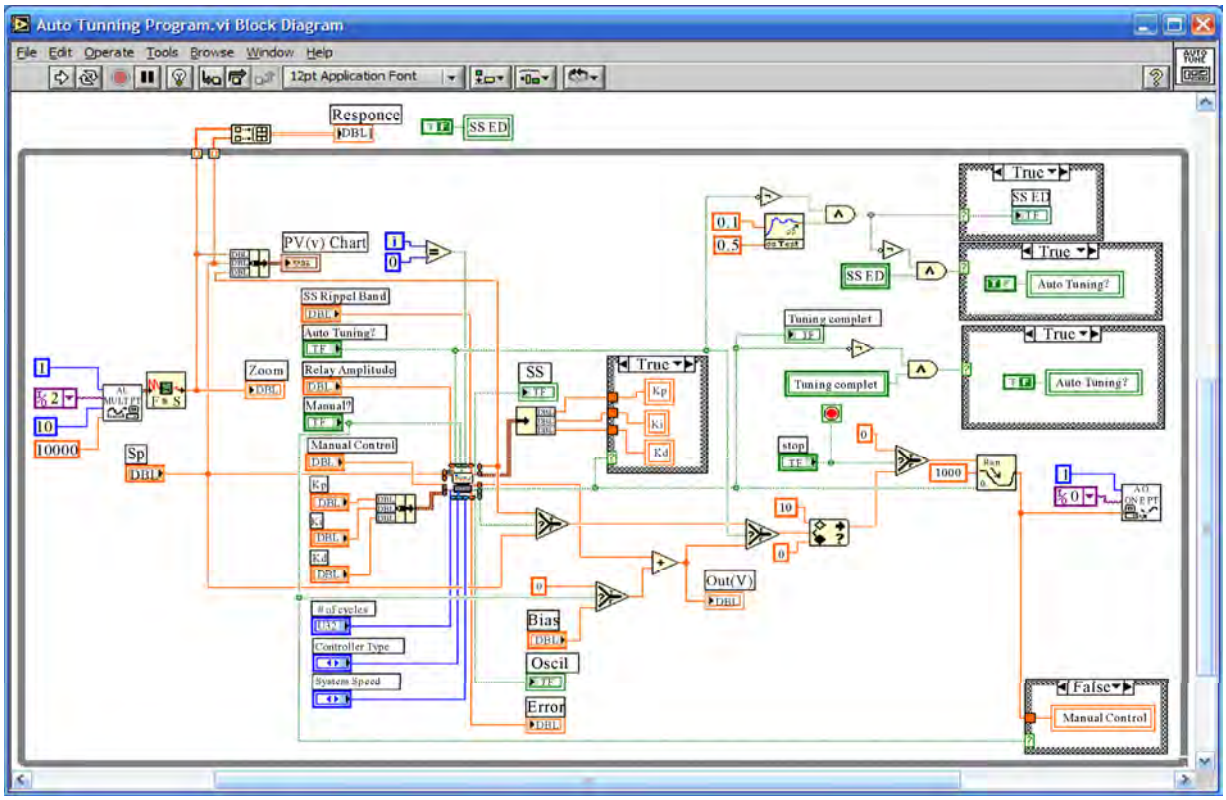


Figure 4. PID auto-tuning VI block diagram.

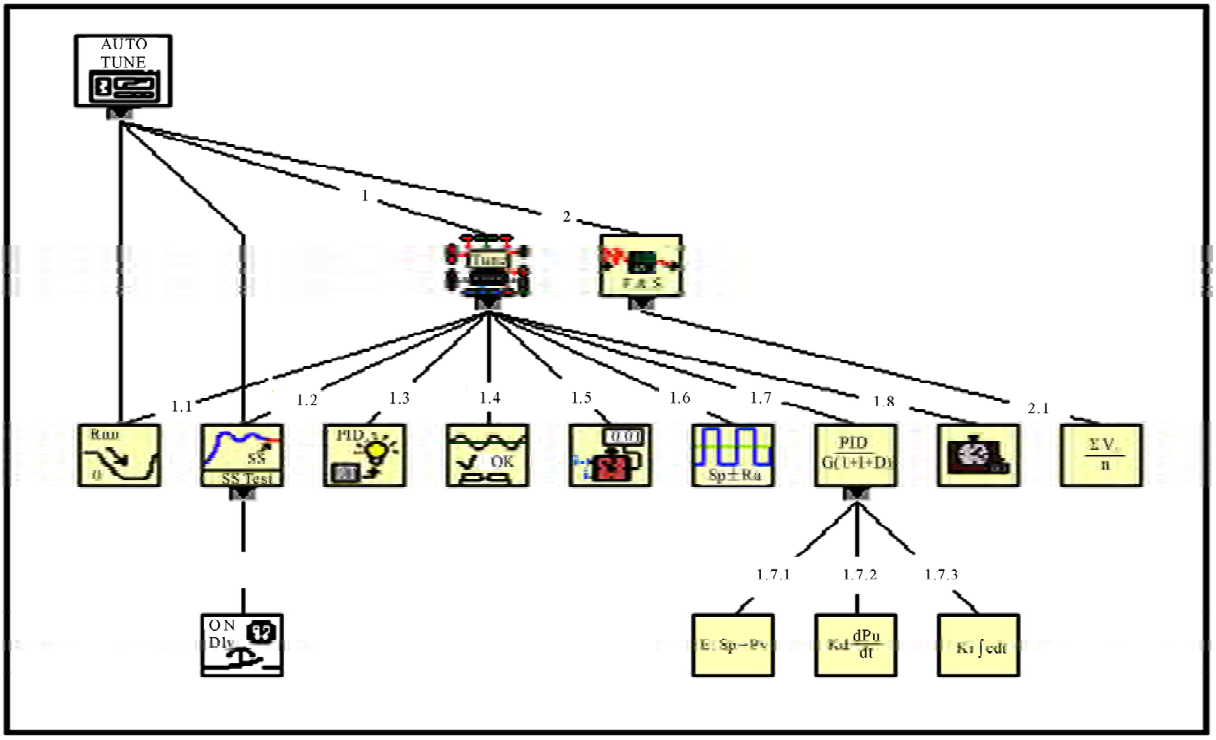


Figure 5. PID auto-tuning VI hierarchy.

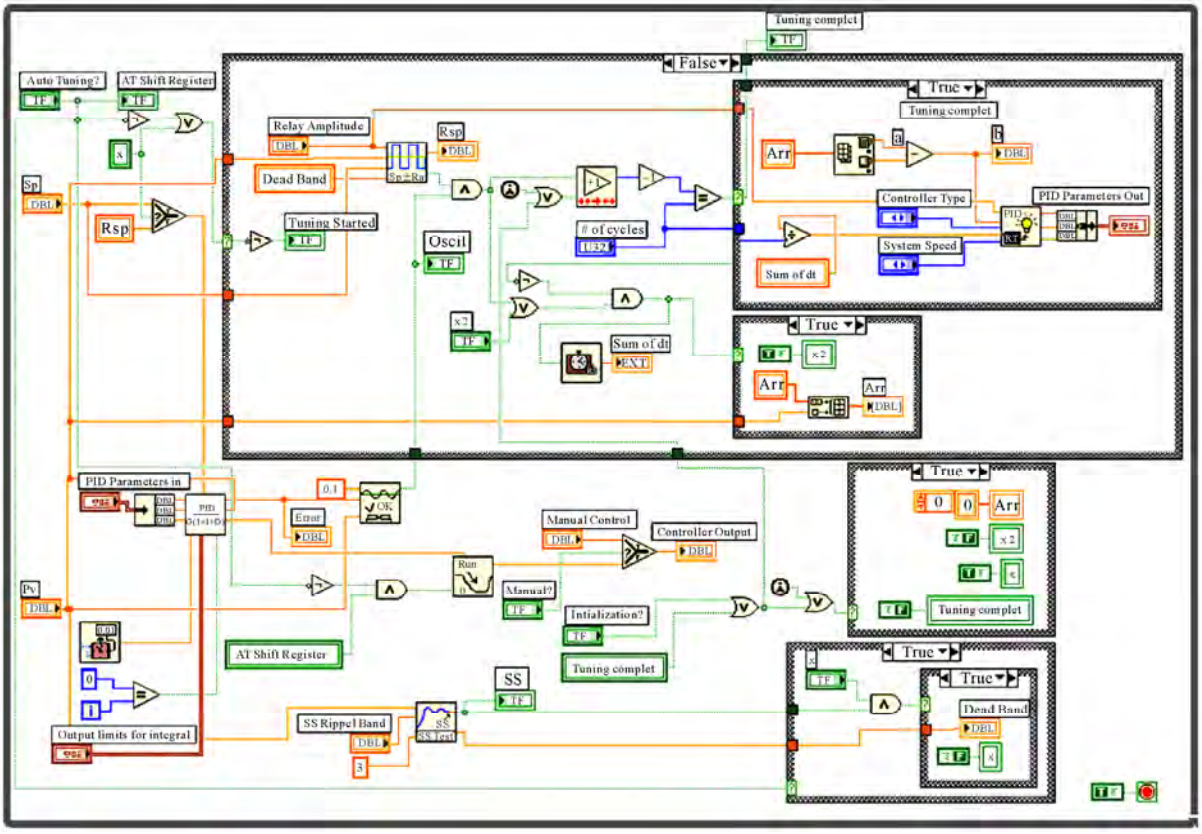


Figure 6. PID auto-tuning (all features) block diagram.

4.2.2. Steady State Test SubVI

It indicates when the (PV) is at steady state. The block diagram is shown in **Figure 8**.

4.2.3. Calculate Best PID Parameters VI

The block diagram is given in **Figure 9**.

4.2.4. Check Oscillation SubVI

It indicates if the (PV) is in an oscillatory mode.

The block diagram is shown in **Figure 10**.

4.2.5. Find Iteration Time SubVI

It calculates the time interval between every two iterations. The block diagram is shown in **Figure 11**.

4.2.6. SetPoint Relay Test SubVI

If PV < (SP − 0.5 dead band), then (SP = SP + relay amplitude) else if PV < (SP + 0.5 dead band), then (SP = SP —relay amplitude). The block diagram is shown in **Figure 12**.

4.2.7. PID Controller SubVI

It performs the standard PID algorithm. The block diagram is illustrated in **Figure 13**.

This SubVI includes:

- Calculate Error SubVI
- Integral term SubVI
- Derivative term SubVI

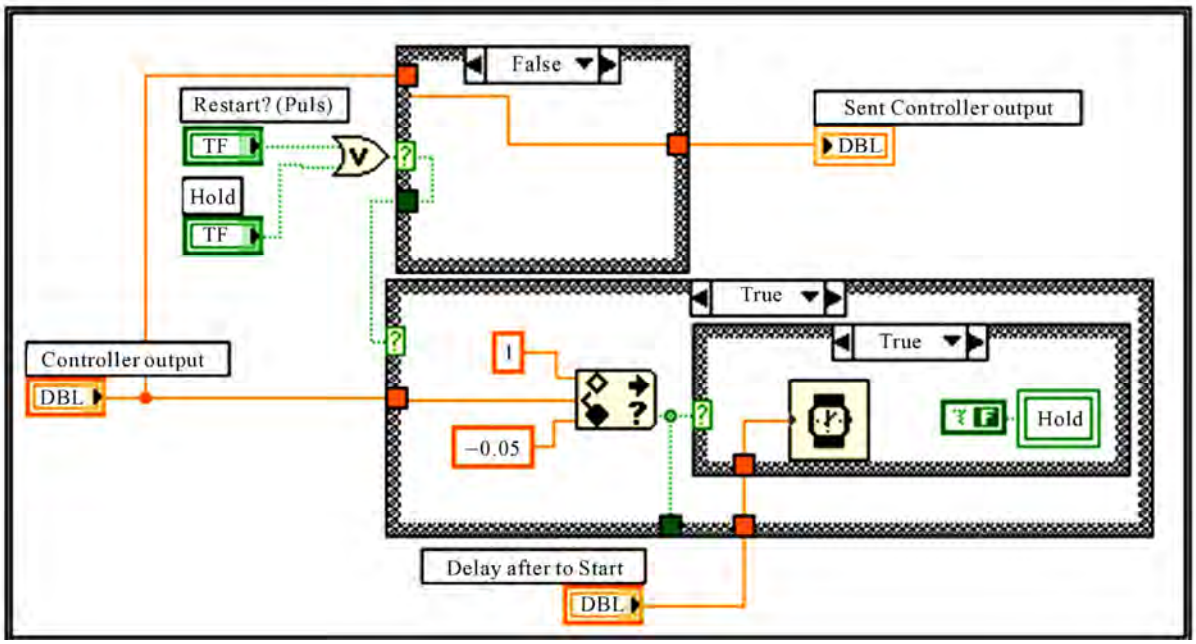


Figure 7. Restart response block diagram.

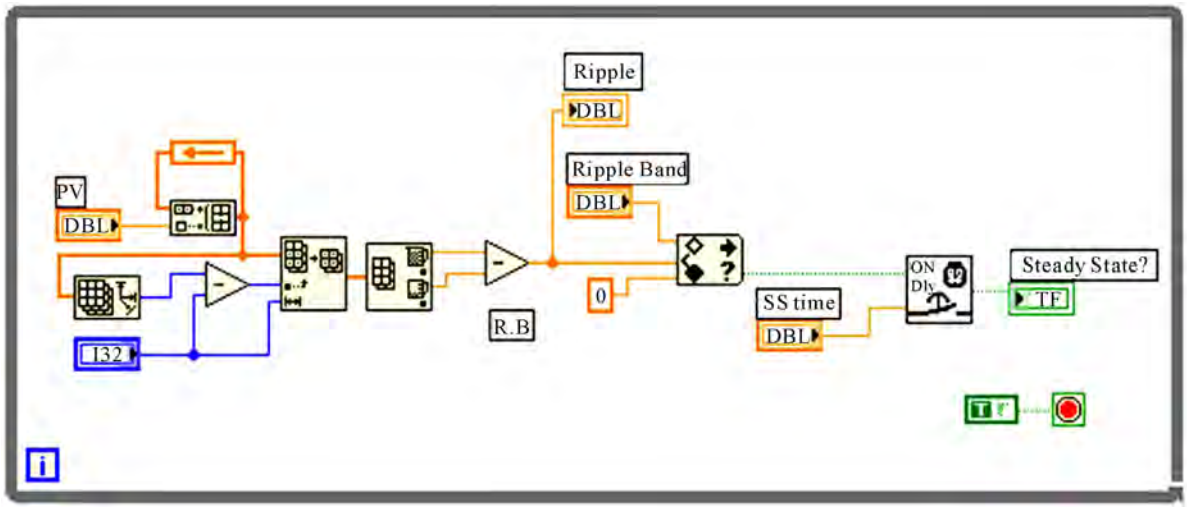


Figure 8. SS test block diagram.

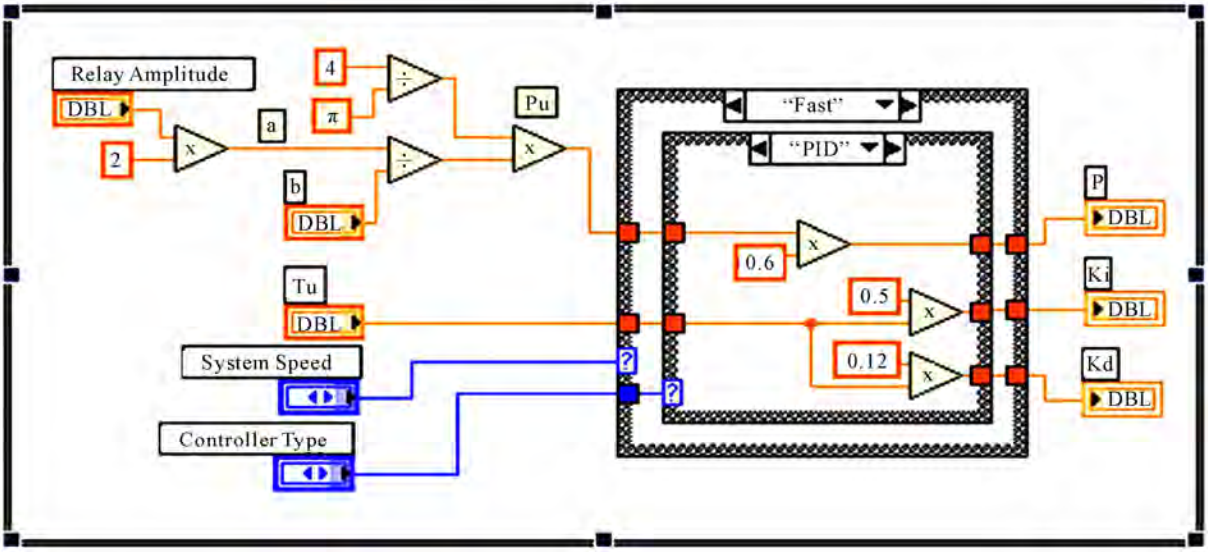


Figure 9. Calculate best PID parameters block diagram.

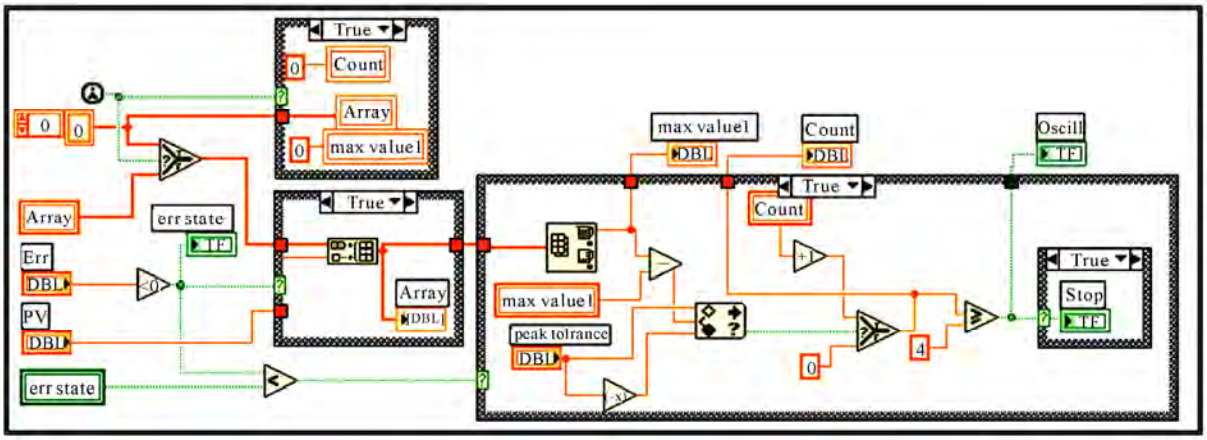


Figure 10. Check oscillation block diagram.

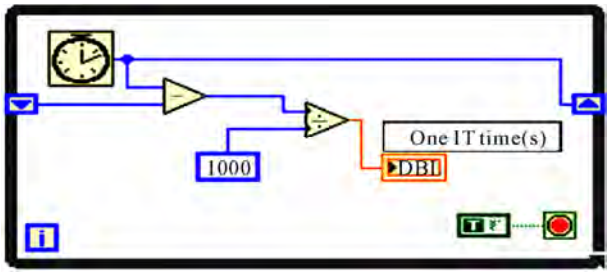


Figure 11. Find iteration time (dt) block diagram.

4.2.8. Stop Watch SubVI

The block diagram is illustrated in **Figure 14**.

4.3. Filtering and Smoothing SubVI

The VI performs double filtering for the read samples in one iteration (acquisition). Firstly it performs an averaging to improve the measurement, then uses (FIR) filter in order to eliminate noise effect on samples. The block diagram is shown in **Figure 15**.

4.3.1. Waveform for Values Average SubVI

The block diagram is given in **Figure 16**.

5. Experimental Procedure and Experimental Results

Initially, the designed PID self-tuned controller was installed. The hardware DAQ-board (PCI-MIO-16E-1) was also installed and the required input/output are configured. The feedback signal and the DAQ-board output were connected to a flow controller process trainer [10-11]. GUI software was designed and used to select the control mode, PID gain values. GUI allows the operator to run manual or self-tuning modes of operation.

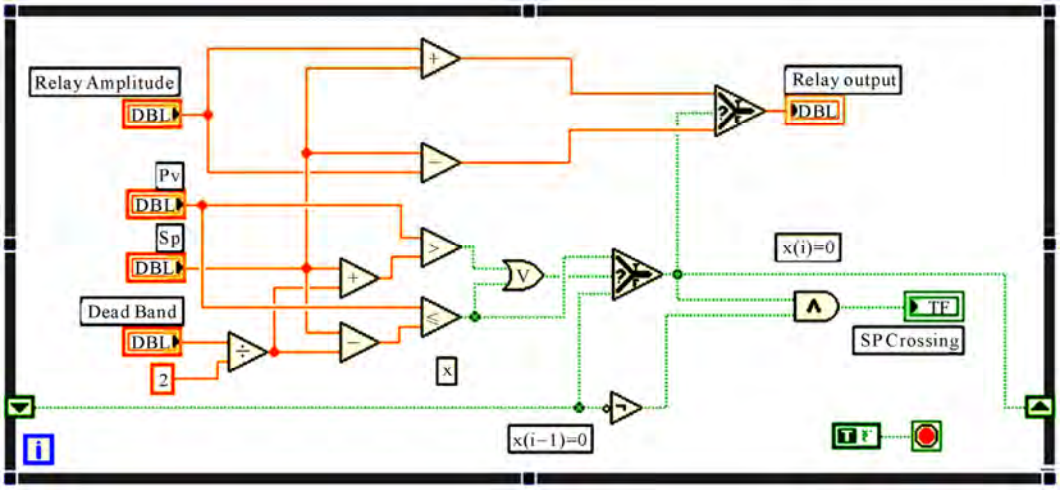


Figure 12. SP relay block diagram.

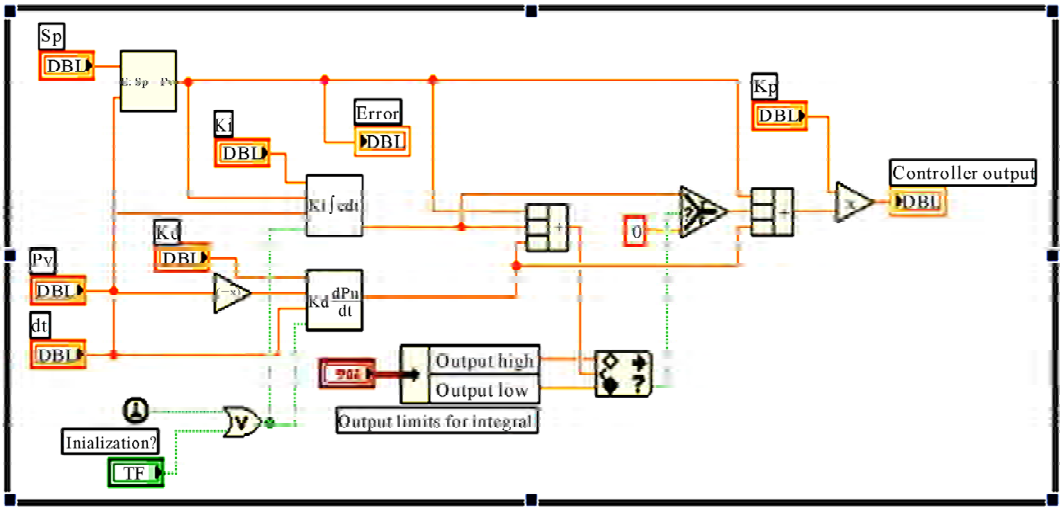


Figure 13. PID controller block diagram.

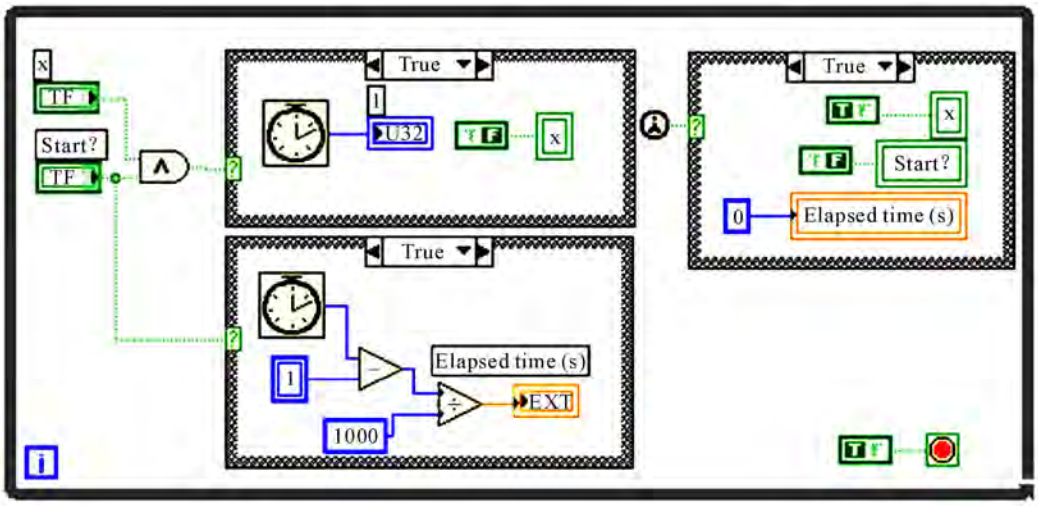


Figure 14. Stop watch block diagram.

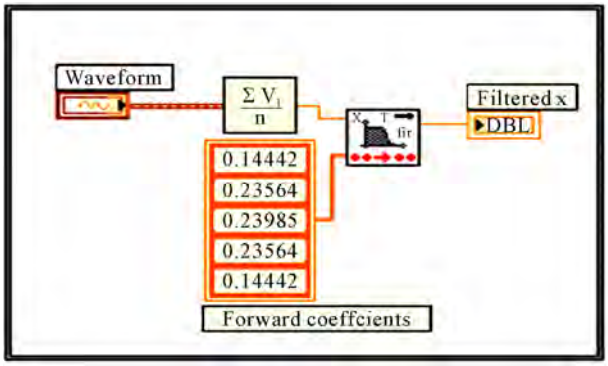


Figure 15. Filtering & smoothing waveform block diagram.

5.1. Proportional Control Mode

Initially the process is kept running in a proportional mode until it reaches a steady state. The user pushes the auto-tuning pushbutton which triggers the tuning process. The system switches to an open-loop mode. When it exists in a steady state the relay test starts. When the process variable oscillations are steady, the gain is calculated and then the controller gain is updated, and the process is run in a closed-loop mode.

Figure 17 illustrates the tuning steps when:

Set point = 4.00 V, K_p = 1.39, relay voltage 1.00 V and the number of oscillations = 4.

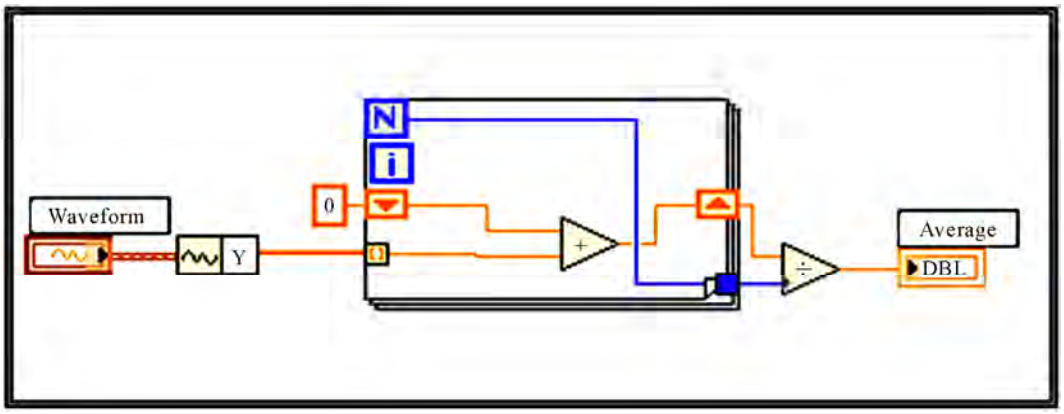


Figure 16. Waveform values average block diagram.

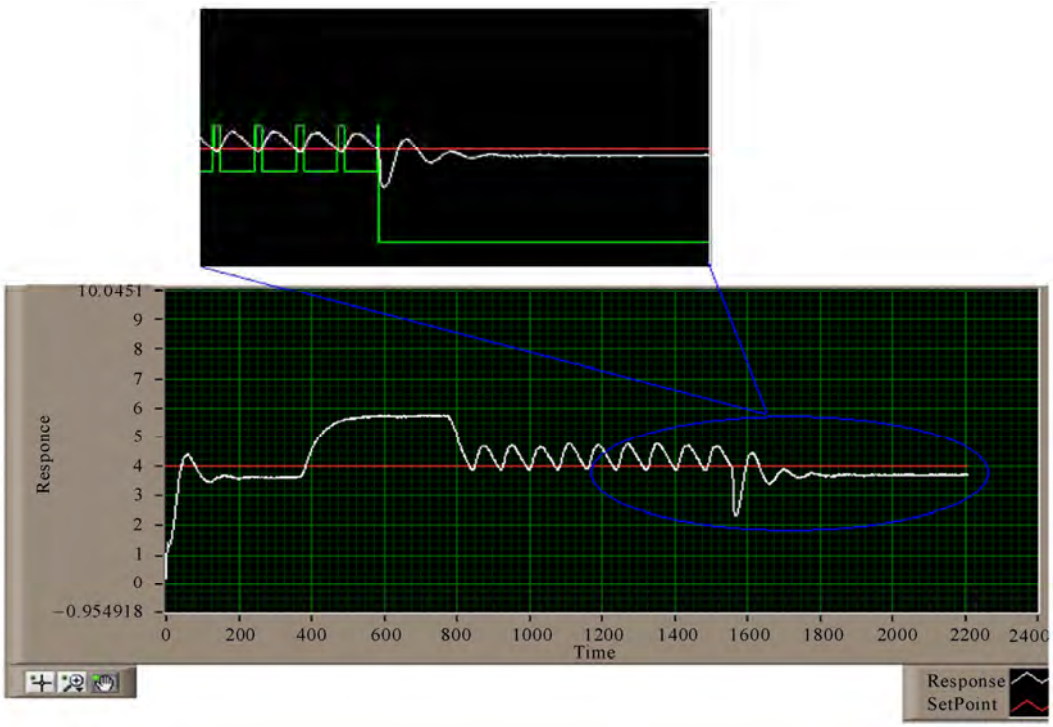


Figure 17. A graph indicator for a P mode.

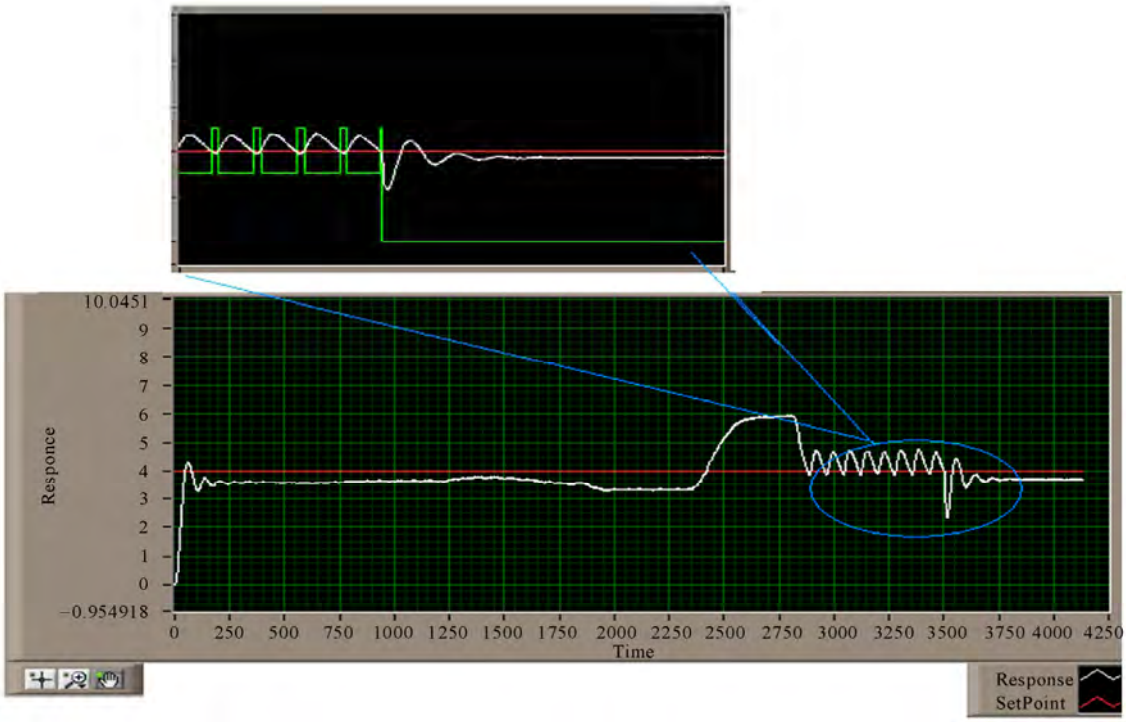


Figure 18. A graph indicator for P mode.

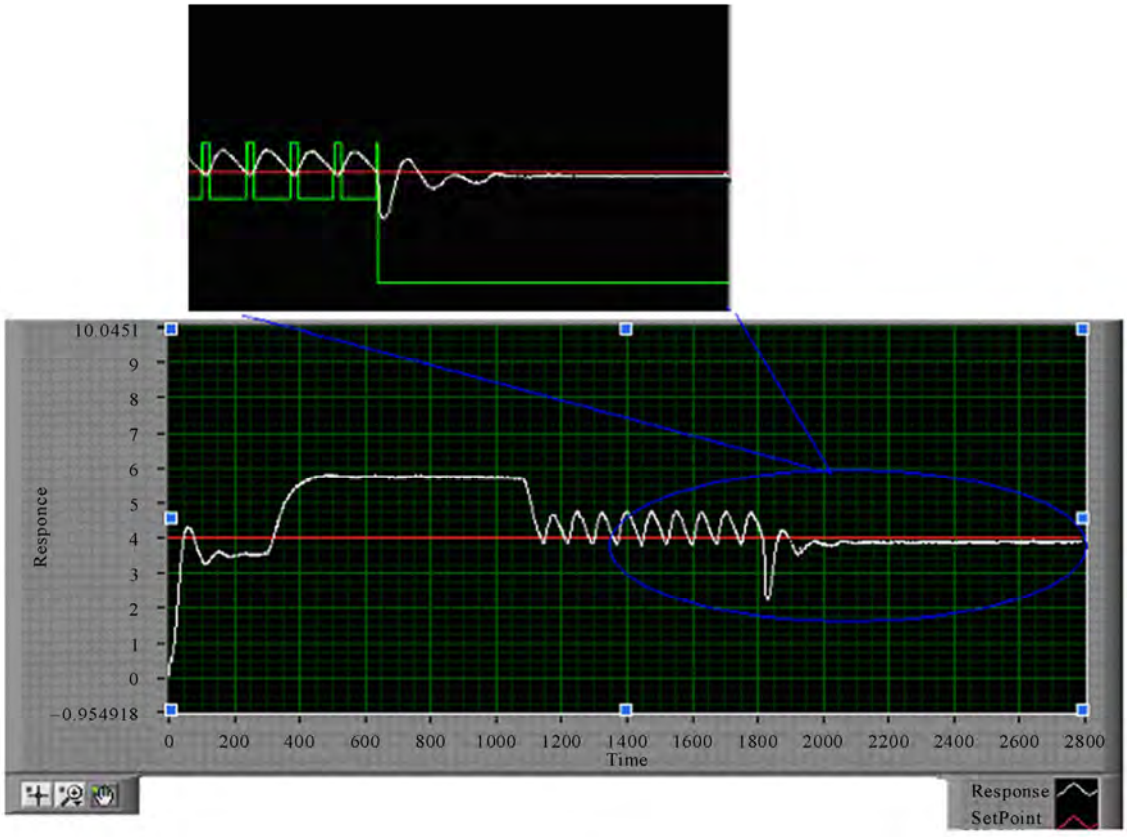


Figure 19. A graph indicator for PI mode.

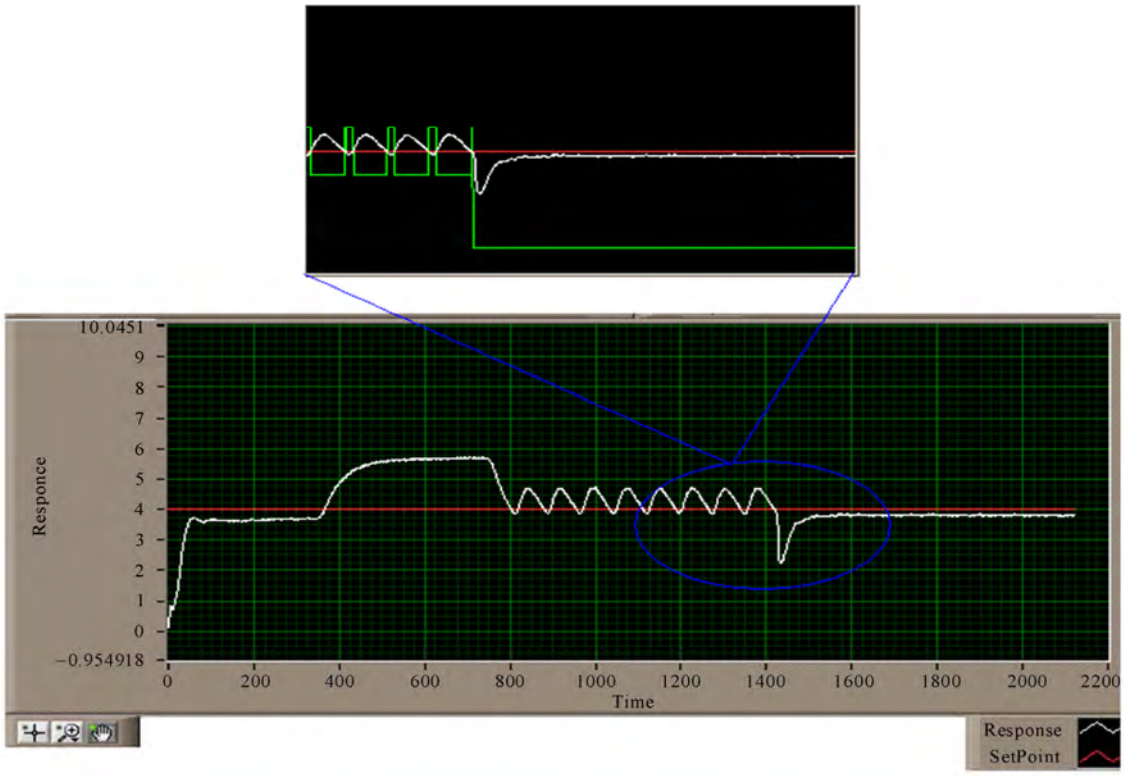


Figure 20. A graph indicator for PID mode.

Figure 18 shows a self-tuning process, when the system was subjected to a disturbance from the load side. The tuned gain value was found (K_p = 1.42).

5.2. PI and PID Control Modes

Tuning steps are the same as in proportional control mode. **Figure 19** shows the tuning for a PI controller when Set point = 4.00 V, K_p = 1.09, K_I = 0.67.

Figure 20 shows the tuning process of a PID controller when: Setpoint = 4.00 V, K_p = 1.77, K_I = 0.83 and K_d = 0.19.

6. Conclusions

Using LabVIEW software a self-tuning PID controller was designed and tested to control the water flow rate. The designed controller may be considered as a development to auto-tuning toolkit of labVIEW.

The controller includes a standard PID controller with the required SubVIs which enables an open-loop self-tuning process without operator intervention. A manual tuning option is also available.

REFERENCES

[1] "Advanced Features in PID Tuning Development Library 2006," National Instruments USA.

[2] "Improving PID Controller Performance," Development zone National Instruments USA.

[3] V. Van Doren, "Fundamentals of Self-Tuning Control," *Control Engineering*, Vol. 54, No. 7, July, 2007.

[4] V. Van Doren, "Applications of Self-Tuning Control," *Control Engineering*, Vol. 54, No. 9, July, 2007.

[5] V. Van Doren, "Auto-Tuning Using Ziegler-Nicholas," *Control Engineering*, Vol. 53, No. 10, 2006.

[6] D. I. Wilson, "Relay-Based PID Tuning," *Automation and Control*, Auckland University of Technology, New Zeland, February-March 2005, pp. 10-11.

[7] K. J. Astrom and Wittenmark, "Adaptive Control," Addison-Welsey Publishing Company Inc., Massacusttes, USA.

[8] G. W. Johnson, "LabView Graphical Programming," MCGarw-Hill, Inc., 2006, New York.

[9] "PID Contol Toolkit Manual," National Instruments, 2006. http://www.ni.com.

[10] "Flow Rate and Level Control Trainer (with DDC Controller under CASSY Lab)," LEYBOLD DIDACTIC GMBH, LEYBOLD DIDACTIC Company, Germany.

[11] E. D. Bolat, Y. Bolat and K. Erkan, "Temperature Control Using Improved Auto-Tuning PID Control Methods," Kocaeli Unversity, Istanbul, 1999.

