PID CONTROLLER DESIGN FOR ASYMMETRICAL TEMPERATURE CONTROL

MEHMET SAİD YÜKSELTEN

BOĞAZİÇİ UNIVERSITY

2010

PID CONTROLLER DESIGN FOR ASYMMETRICAL TEMPERATURE CONTROL

by

Mehmet Said YÜKSELTEN

B.S., Electrical&Electronics Engineering, Yeditepe University, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Systems and Control Engineering
Boğaziçi University

2010

PID CONTROLLER DESIGN FOR ASYMMETRICAL TEMPERATURE CONTROL

APPROVED BY:

Prof. Eşref Eşkinat (Thesis Supervisor)	. . . . .
Prof. Mehmet C. Çamurdan	. . . . .
Assoc. Prof. Yağmur Denizhan	. . . . .

DATE OF APPROVAL: 26/04/2010

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Prof. Dr. Eşref EŞKİNAT who helped to me during completing the project.

I have also benefited from the discussions with my friends and so I would like to thank to them, especially to Ertan BALABAN.

ABSTRACT

PID CONTROLLER DESIGN FOR ASYMMETRICAL TEMPERATURE CONTROL

In today's industrial environments, there is an increasing number of modelling and controlling researches of the processes which are used in many types of applications. However, there are only some limited researches on asymmetrical processes. Temperature control system is a common example of those processes. Some examples to that are industrial oven and furnace, air conditioned mediums, medicine and dye factories, food industry, heating and cooling systems and so on. The asymmetrical processes are the systems whose dynamics may change depending on the direction. In those systems, since the powerful heaters, the heating process realizes rapidly but that the cooling process slowly. That's why, the classical PID control method may not be enough to get the desired results.

On this project, we developed a new model by firstly reducing the overshoot and minimizing the settling time with Full power and PWM modes. Then, we made some improvements on the standard PID method to get a system output like symmetrical behaviour.

In order to test the applied model, we developed a full control system with a microcontroller based embedded board and a user interface on PC to be controlled by the user. The system mainly consists of a heating rezistance, thermocouple, microcontroller based board and a PC to visualize the system.

In this project, firstly the general characteristics of the asymmetrical processes are discussed and then a new control model is presented with some complicated methods. After talking about the software and the hardware models, the results are discussed.

ÖZET

ASİMETRİK SICAKLIK KONTROLÜNDE PID KONTROL TASARIMI

Günümüz endüstrisinde pek çok uygulama tarafından kullanılmakta olan proseslerin modellenmesi ve kontrol edilmesi üzerine yapılan araştırmalarda önemli bir artış gözlemlenmektedir. Ancak asimetrik proses kontrolünde yapılan sınırlı sayıda araştırma mevcuttur. Sıcaklık kontrolü bu proseslerin sık kullanılan bir örneğidir. Buna örnek olarak endüstriyel ocak ve fırın, klima ortamları, plastik enjeksiyon baskı makineleri, ilaç ve boya fabrikaları, yemek endüstrisi, ısıtma ve soğutma sistemleri gibi uygulamaları verebiliriz. Asimetrik sistemler, sistem dinamikleri yöne göre değişen sistemlerdir. Bu sistemlerde güçlü ısıtıcılar dolayısıyla ısınma hızlı olurken, soğuma daha yavaş gerçekleşmektedir. Bununla birlikte, sistemin belirlenen hedef değerine oturması gerekmektedir. Bu sebeple klasik PID kontrol metodu istenen hedeflere ulaşmada yeterli olamamaktadır.

Bu projede, bir sistem üzerindeki asimetrik etkiyi ortadan kaldırmak üzere, tam güç ve PWM modları yardımıyla ilk olarak, aşımı engelleyip oturma zamanını mimimuma indirdik. Sonrasında sistemin kararlı-hal bölgesinde çalışarak, simetrik davranışa sahip bir sistem çıkışı elde etmek üzere PID kontrol metodu üzerinde bazı iyileştirmeler yaptık.

Uygulanan modeli test etmek için, mikrodenetleyici kontrollü bir tümleşik devre ve kullanıcı tarafından PC üzerinden kontrol edilmek üzere bir kullanıcı arayüzüne sahip tam donanımlı bir kontrol sistemi tasarladık. Sistem ana olarak bir ısıtma rezistansı, termokuple, mikrodenetleyici temelli bir devre ve görüntüleme amacıyla bir PC'den meydana gelmektedir.

Bu projede ilk olarak asimetrik proseslerin genel karakteristikleri anlatıldıktan sonra bir çok metottan meydana gelen yeni bir kontrol modeli sunulmaktadır. Sonrasında projenin yazılım ve donanım modelleri açıklanmakta ve sonunda sonuçlar tartışılmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

a	Amplitude of the system output cycle
A^+	Areas of the positive cycles of the relay output
A^-	Areas of the negative cycles of the relay output
D	Derivative
e	System error
e(t)	Input signal to the nonlinear element
G(s)	Transfer function
h	Amplitude of the relay output cycle
I	Integral
k	Compensation constant
K_d	Derivative gain
K_i	Integral gain
K_p	Proportional gain
K_u	Ultimate gain
P	Proportional
P_u	Ultimate period
r	System input value
t_1	Rising time
t_2	Landing time
u	Controller output signal
V_{ref}	Reference voltage
w_u	Ultimate frequency
y	System output
Δa	Biased signal
δ	Degree of the asymmetry
ADC	Analog to Digital Converter
PID	Proportional-Integral-Derivative
PWM	Pulse Width Modulation

1. INTRODUCTION

1.1. Statement of The Problem

In today's industrial environments, there is an increasing number of modelling and controlling of the processes which are used in many types of applications. However, there is not enough research on the asymmetrical processes.

Asymmetrical processes consist of two modes such as heating and cooling whose dynamic characteristics are very different. The different behaviour of the same system in two modes causes it in difficulty to be controlled. So, one directional control may not be suitable for such systems.

Temperature control systems are common examples of those processes. Some examples of temperature control applications are industrial oven and furnace, air conditioned mediums, plastic injection pressure machines, medicine and dye factories, food industry, heating and cooling systems and so on. As an example to those systems, a heating resistance has faster dynamics during the heating state than the cooling. Therefore a control application based on the heating state will not work for the cooling state.

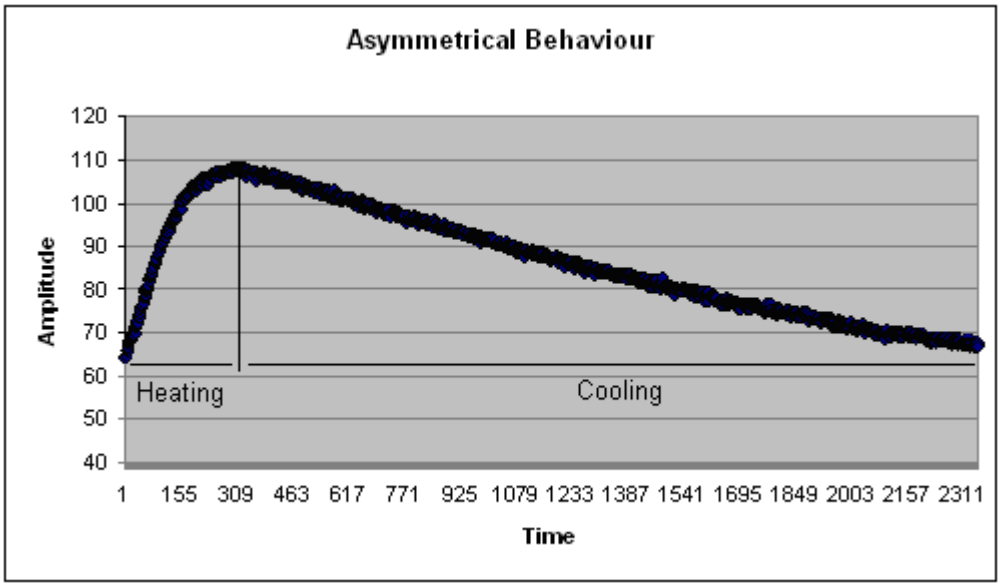

Figure 1.1. Asymmetrical behavior of a heating resistance during heating and cooling modes

This difference becomes more important if there is no active cooling system in the medium. Because of the powerful heaters, the heating process occurs rapidly but that the cooling process is slow as shown in Figure 1.1. They also need the setpoint to be set. That's why, the classical PID controller may not be enough to get the desired results and the engineers have been working to find some self-regulated methods against the asymmetrical problems.

1.2. Previous Approaches

PID controller is a common practice in regulation of the asymmetrical processes. However, if the degree of the asymmetry is severe, then some additional methods become necessary to handle with the situation. In this case, researchers add some more dynamics on the standard PID controller to make the system behaving like symmetrical.

Gain-Scheduled control method adds a self-organising gain and a delay element on the standard relay tuning method to have symmetrical behaviours on two modes of the system. In order to succeed this criterion, the approach aims to have the same ultimate gain and ultimate frequency on both modes by applying a cascaded function. It considers the amplitudes of the positive and the negative cycles to calculate the gin factor and derive the function equation. On the other hand, the approach considers the duty cycle of the negative cycle and the period of the system to derive the delay element. Finally, the proposal affects the system dynamics on two modes and succeeds a more symmetrical behaviour at the system output.

Use of Biased – Relay Feedback method works on the asymmetrical oscillations in feedback systems by defining two points on the Nyquist curve at frequencies. The method offers new approaches for the steady-sate gain, K_p. The method then applies two model structure. Model 1 determines the deadtime D over K_p which is already known while Model 2 determines the deadtime from the observation. Then new equations are offered for each model to calculate the ultimate period.

1.3. Proposed Work

Temperature control systems are the most widely used asymmetrical processes in the industry. Generally, those systems have the severe degree of asymmetry during heating and cooling modes. Therefore, they are very convenient systems to observe and work on the asymmetrical structure with also being so complex, on the other hand.

During our study on this problem, we aim to observe the asymmetrical structure of a system and improve it for different behaviors on two modes by developing new methods. In order to test the methods, we designed an experimental temperature control product for the project. The product is managed by a microcontroller on it and has a temperature resistance as the heater which is heated by electrically controlled by a relay and a J-type thermocouple is used as the temperature sensor. The relay is controlled by the signals from a PC. We have designed a special user interface on PC to plot the system output on the graph and fully trace the system over that. The communication has been realized over USB or serial port (RS-232).

During the experiments, we aim to handle with the problems with asymmetrical systems such as the dead band characteristic, huge overshoot, many oscillations around the setpoint and so long settling time.

In the proposal section, we realize a 4-mode tuning algorithm and apply it on the system. The modes are Full Power, PWM, G(s), and PID with variable K_i.

In Full Power Mode, we aim to cancel the dead band effect on the system. So, we tune the system by applying full power up to a given temperature value and repeated it for a few different suitable points and then note the final results. This mode is to cancel the dead band effect especially for the initial part of the control in order to get a minimized overshoot while speeding up the system.

PWM Mode starts just after Full Power Mode. During this mode, we apply a constant PWM signal as the input to the system. By the way, it supports Full Power Mode and helps the system to settle on the setpoint smoothly. After Full Power Mode, without

PWM mode, the system waits for a while so that it is driven by the energy loaded during the deadband region. However, it again gets into the deadband region and causes an offset by undershooting with increasing the settling time. PWM mode aims to improve the system against these problems and sets up a supporting bridge between Full Power and PID modes so that the system is kept loaded just after Full Power Mode and by the way it cancels the offset.

In G(s) mode, we apply the relay autotuning method to get the the PID parameters and then apply the bilinear transform for the digital system. Together with this operation, we also have some information about the asymmetrical structure of the system and apply it on K_i to improve the asymmetrical structure.

Finally, we have apply PID Mode which is the most effective and innovative mode. The parameters which are tuned in G(s) mode are used during the control strategy but also applying some new techniques. Setting integral term is one of those improvements which loads K_i by some amount which makes the integral part of PID afford to drive the system to the set point. This is applied when the current temperature reaches to the 10 per cent below the setpoint. It is to fire the system again to go through the set value.

As a new novel approach to the standard PID, we apply a changing K_i during heating and cooling modes against the asymmetrical structure. We define a new K_i constant, k, retrieved from the slope of the system during heating and cooling based on the time durations. The algorithm works with this improvement by changing the K_i parameter accordingly. The aim is to minimize the asymmetrical effect on the system especially in terms of the steady-state errror by loading the system by different amounts in two heating and cooling modes.

The experiments of the combined modes are finally realized and the results are discussed.

2. LITERATURE SURVEY

2.1. PID Controller

2.1.1. Introduction to PID Controller

Proportional-Integral-Derivative (PID) controllers are control algorithms commonly found in industrial applications. Because of their robust performance, simplicity of structure and easy comprehension in principle, PID controllers are extensively used in process industries. [1]

The PID Controller is an improved and complex method which have been used for many years. In this method, there are three control parameters to achive the best performance. They help the system to compensate for changes in the system. The proportional, the integral and the derivative terms must be individually tuned based on a particular system. In the method, the proportional parameter results a faster response, the integral parameter eliminates the steady state errors and the derivative parameter reduces the transient errors and decreases the overshoot. Totally, it provides the most accurate and stable control of the overall system. The method considers the error signal which is the difference between the setpoint and the actual point. During the control process, the proportional works only with the current error, the integral works with the total sum of the error signals and the derivative works with the change in the error signals.

The controller includes three constant parameters to control the system; the proportional, the integral and the derivative values. Each parameter results a different effect on the system: The *proportional* term acts on the current error, the *integral* term considers the sum of recent errors, and the *derivative* term determines the reaction based on the rate at which the error has been changing. The combination of all these actions adjusts the process over an intermediate plant such as the power supply of a heating resistance.

The values of PID parameters may vary according to target system. So, they must be tuned according to the nature of that system. By tuning these constants, the controller will able to result a more appropriate control output designed for specific process requirements. The quality of the controller can be evaluated in terms of the responsiveness of the controller to an error which is the degree of the system overshoot the setpoint and the degree of system oscillation.

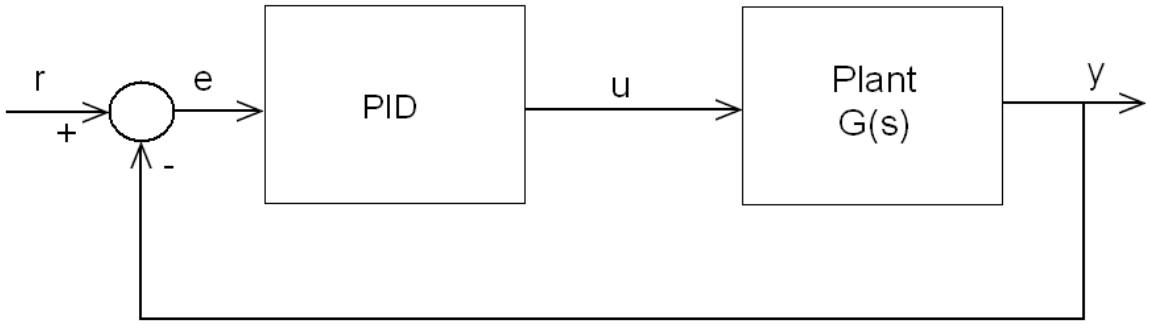

Figure 2.1. The PID control system

Getting all the three parameters into the consideration is not a mandatory issue for all the systems. Some processes may require using only one or two modes to provide a sufficient system control. In this situation, for the simplicity of the system, this application can be achieved by setting the undesired parameters to zero. However, for some applications, each parameter may improve the system by different capability. The controller maybe called as PI contoller in the absence of the derivative term and similarly PD, P or I controller accordingly. For many processes, the error change is smaller which means the system output changes slowly. Therefore, in this case, the derivative term acts only a small or no effect on the system. So, PI controllers are particularly common for many processes in the industry.

A PID controller gets the error value as the difference between the current measured process output and the desired setpoint. The controller aims to minimize the error by adjusting the control parameters.

2.1.2. PID Tuning Methods

2.1.2.1. Manual Tuning. It would be better to first consider the manual tuning of the PID parameters to understand the meaning of each one in practice. At this point, the order of the parameters to be tuned is important. Therefore, a more useful method is to set K_i and K_d values first to zero. Then, set an initial value to K_p and increase it until which the system oscillates. K_p should be set to approximately half of this value. Then increase K_i until any offset is correct in sufficient time for the process. Notice that large K_i will cause instability, so this parameter should be adjusted sensitively. Finally, increase K_d until the output is sufficiently rapid to reach the setpoint after a load disturbance. Again, large K_d may cause an excessive response and overshoot. A quick PID control loop tuning cause overshoots that is what to destroy. In order to eliminate the overshoot, K_p is required to be significantly less than half that of the K_p setting causing oscillation.

2.1.2.2. Ziegler-Nichols Tuning Rule. The most well known tuning methods are those developed by Ziegler and Nichols. They have had a major influence on the practice of PID control for more than half a century. The methods are based on characterization of process dynamics by a few parameters and simple equations for the controller parameters. It is surprising that the methods are so widely referenced because they give moderately good tuning only in restricted situations. [2]

Ziegler and Nichols developed a method for determining the values of K_p, K_i, and K_d based on a simple characterization of the frequency response of the process dynamics. They realized numerous experiments and proposed rules based on the transient step response of a plant.

The method is based on a point where the Nyquist curve of the transfer function intersects the negative real axis. This point is then pointed by the frequency, w and the gain, K at that frequency. Finally, the point is called the ultimate point indicated by the parameters K_u, $\frac{1}{K}$ and T_u, $\frac{2\pi}{\omega}$ which are called the *ultimate gain* and the *ultimate period*.

Table 2.1. Controller parameters for Ziegler-Nichols frequency response method

Controller	K_p	T_i	T_d
P	0.5 K_u		
PI	0.45 K_u	0.833 T_u	
PID	0.6 K_u	0.5 T_u	0.12 T_u
PID some overshoot	0.33 K_u	0.5 T_u	0.33 T_u
PID no overshoot	0.2 K_u	0.3 T_u	0.5 T T_u

In order to calculate the parameters, set them to cause the control action be proportional.Then increase the gain until the output oscillates. At this point , the gain is noted as K_u and the period of the oscillation is T_u. Finally, the controller parameters can be calculated by the equations shown in Table 2.1.

Ziegler and Nichols have also proposed another method, called S-shaped reaction curve. It can be applicable for the systems whose unit-step response resemble an S-shaped curve with no overshoot.

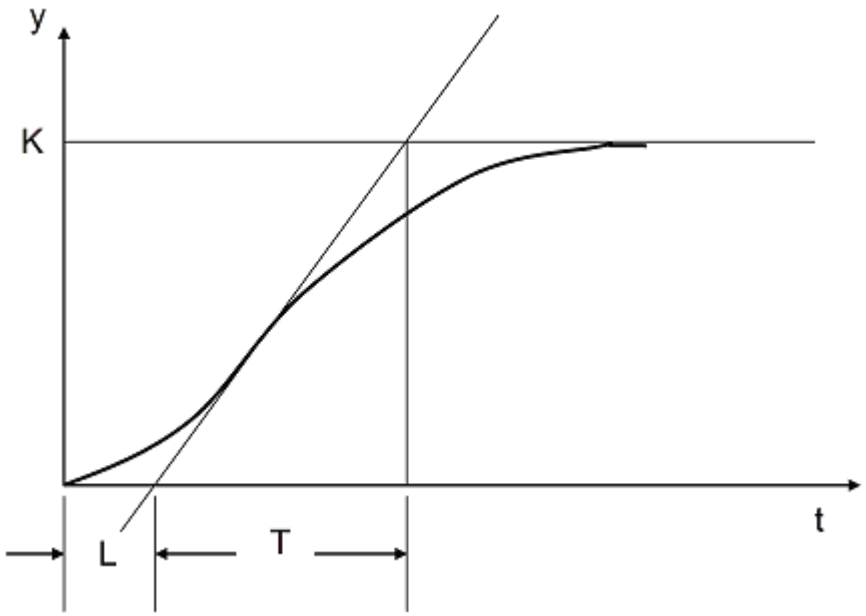

Figure 2.2. Step response of the Ziegler-Nichols S-shaped reaction curve method

The S-shaped reaction curve can be indicated by two constants, the delay time L and the time constant T, which are the intersections of the tangent line with the time axis and the steady-state level line.

The static gain (K) is obtained from the final steady-state level of the process output. Remember that the process output must be scaled with the change in the control variable. The intercept of the tangent to the step response that has the largest slope with the horizontal axes gives L. [3]

Then, the PID parameters can be calculated. The calculation results can be seen in Table 2.2.

These settings result a response with an overshoot of 25 per cent and a good settling time. However, the values are not the optimal ones and need to be additionally fine tuned for a better performance.

Table 2.2. Controller parameters for Ziegler-Nichols S-shaped reaction curve method

Controller	K_p	K_i	K_d
P	$\frac{T}{L}$	0	0
PI	$0.9\frac{T}{L}$	$0.27\frac{T}{L^2}$	0
PID	$1.2\frac{T}{L}$	$0.6\frac{T}{L^2}$	$0.6T$

2.1.2.3. Relay Method. The Relay method is an improved version of the Ziegler-Nichols method. The method is based on the ultimate gain, K_u and the ultimate period, T_u. The basic idea is that many processes get into the limit cycles under relay feedback control. You can see the block diagram of the relay feedback control method in Figure 2.7.

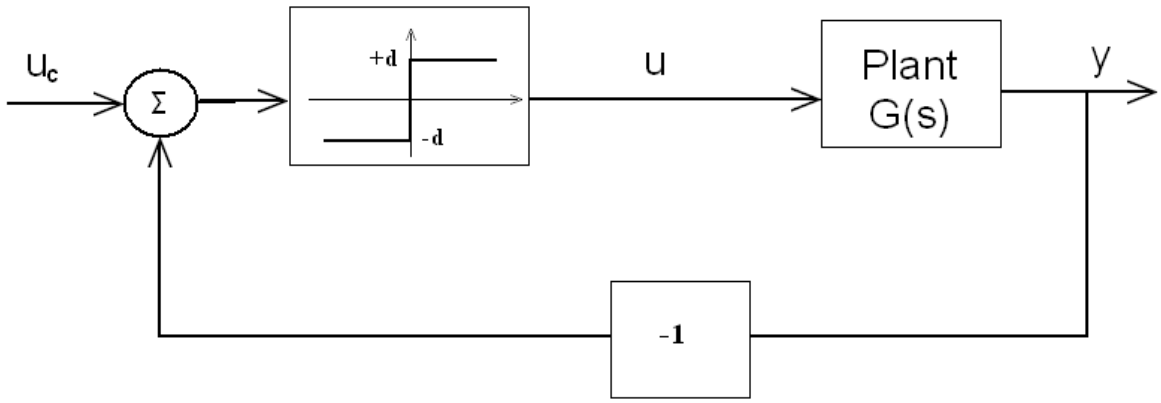

Figure 2.3. Block diagram of the relay feedback control method [4]

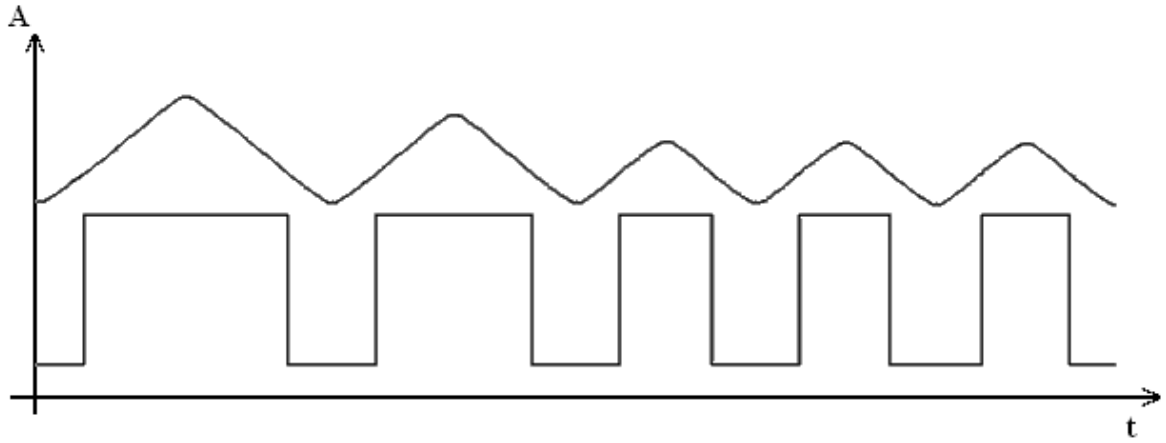

Figure 2.4. Relay method system input (square wave) and output (sinusoidal wave) signals

In order to realize the relay method, you should follow the following procedure.

i. Apply a constant input u_c and get the corresponding steady state output, y.

ii. Then, change the setpoint to y.

iii. Next change the input state with a small value of h.

The algorithm is:

$$u = \begin{cases} u_c + h, e \geq 0 \\ u_c - h, e < 0 \end{cases}$$

iv. Wait for the system to get into the limit cycle.

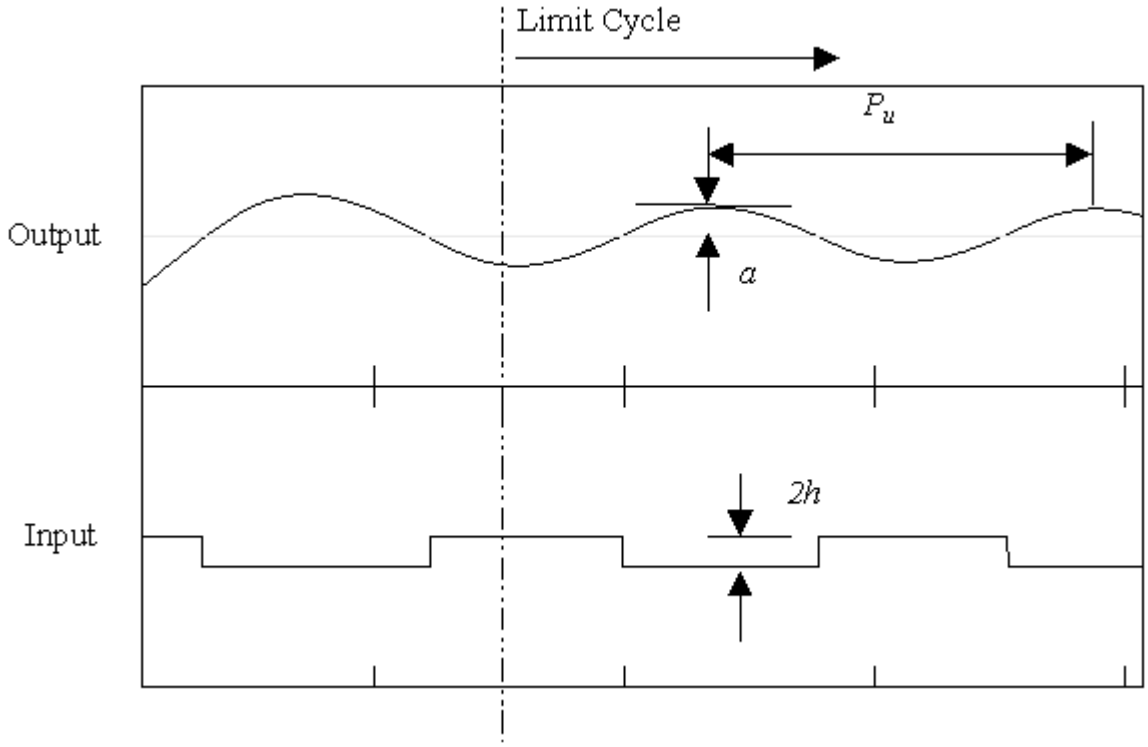

Figure 2.5. Calculation of the Relay method parameters using the system input and the output signals

v. After having the limit cycle response, measure the amplitude of the output cycle, *a* and note the value of *h*.

vi. Now, measure the ultimate period, P_u and calculate the ultimate gain, K_u using the relay parameters:

$$K_u = \frac{4h}{\pi a} \tag{2.1}$$

vii. Finally, use the equations in Table 2.1 to get the resulting PID values.

2.1.3. Discrete Implementation

PID algorithm considers the current, the history and the rate of the input signals. This results a more accurate and stable control method.

In mathematical representation,

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int e(\sigma)d\sigma + T_d\frac{de(t)}{dt}\right) \tag{2.2}$$

If we are talking about the discrete case, we may then approximate the integral and the derivative terms to get the following equations.

$$\int_0^t e(\sigma)d\sigma \approx T\sum_{k=0}^{n} e(k)$$

$$\frac{de(t)}{dt} \approx \frac{e(n)-e(n-1)}{T} \tag{2.3}$$

$$t = nT$$

where n is the discrete step at time t. Then, the Equation 2.2 becomes,

$$u(n) = \mathrm{K}_p \mathrm{e(n)} + \mathrm{K}_i \sum_{\mathrm{k=0}}^{\mathrm{n}} \mathrm{e(k)} + \mathrm{K}_d (\mathrm{e(n)} - \mathrm{e(n-1)}) \tag{2.4}$$

where $\mathrm{K}_i = \frac{\mathrm{K}_p T}{\mathrm{T}_i}$, $\mathrm{K}_d = \frac{\mathrm{K}_p T_d}{\mathrm{T}}$ and T, $\mathrm{T_i}$ and $\mathrm{T_d}$ are the time constants of the proportional, integral and derivative terms.

Finally, rather than working with the control input, the derivative term can be manipulated to work on the process values in order to cancel the undesired effects on the control input.

$$u(n) = \mathrm{K}_p \mathrm{e(n)} + \mathrm{K}_i \sum_{\mathrm{k=0}}^{\mathrm{n}} \mathrm{e(k)} + \mathrm{K}_d (\mathrm{y(n)} - \mathrm{y(n-1)}) \tag{2.5}$$

From the equation, the proportional term is directly proportional with the error and the integral term is the sum of the error signals. Additionally, the derivative term is proportional with the difference of the last two output signals. By the way, the system can easily calculate the necessary parameters and result the final value.

2.1.4. Bilinear Transformation

Bilinear transformation is also known as Tustin's method which is only one way within several mehods to transform s-plane into z-plane. The transformation can be described as:

$$s = \frac{2}{T_s}.\frac{z-1}{z+1} \tag{2.6}$$

where T_s is the sampling period of the discerete-time system. When we substitute the equation into the PID controller, then the discrete-time PID controller becomes:

$$G_c(z) = K_p + K_i \frac{T_s}{2}.\frac{z+1}{z-1} + K_d \frac{2}{T_s}.\frac{z-1}{z+1} \tag{2.7}$$

$$G_c(z) = \frac{\left(K_p + \frac{K_i T_s}{2} + \frac{2K_d}{T_s}\right)z^2 + \left(K_i T_s - \frac{4K_d}{T_s}\right)z + \left(-K_p + \frac{K_i T_s}{2} + \frac{2K_d}{T_s}\right)}{z^2 - 1} \tag{2.8}$$

Now we can convert it approximately into the canonical equation by:

$$G_c(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{p=0}^{M} z^{-q} a_q}{\sum_{p=0}^{N} z^{-p} b_p} = \frac{a_0 + a_1 z^{-1} + + a_2 z^{-2}}{1 + b_1 z^{-1} + + b_2 z^{-2}} \tag{2.9}$$

where

$$a_0 = K_p + K_i \frac{T_s}{2} + K_d \frac{1}{T_s}$$

$$a_1 = -K_p + K_i \frac{T_s}{2} + K_d \frac{-2}{T_s}$$

$$a_2 = \frac{K_d}{T_s}$$

$$b_1 = -1$$

$$b_2 = 0$$

Once we have the z-domain transfer function of the PID controller, we can convert it into the digital time domain by using the Linear constant-coefficient difference equation (LCCD):

$$y(n) = \sum_{q=0}^{M} x[n-q]a_q - \sum_{p=1}^{N} y[n-p]b_p \tag{2.10}$$

$$y[n] = x[n]a_0 + x[n-1]a_1 + x[n-2]a_2 - y[n-1]b_1 - y[n-2]b_2$$

Finally, from this difference equation, we can create a digital filter structure to implement the PID controller.

Calculation of the discrete-time PID contoller in computer systems can be done in many ways. The following matrix equations can be into an m-file and then PID gains are put into that.

$$\text{numerator} = \left[\left(K_p + (K_i \times T_s \times 0.5) + ((2 \times K_d) \div T_s)\right) \quad \left(K_i \times T_s - ((4 \times K_d) \div T_s)\right) \right.$$
$$\left. \left(-K_p + ((K_i \times T_s) \div 2) + ((2 \times K_d) \div T_s)\right)\right] \tag{2.11}$$
$$\text{denominator} = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

Another way may be using the matlab for c2dm function in order to convert continuous-time to discrete-time PID controller without making any algebraic substitution.

$$[\text{den}, \text{num}] = \text{c2dm}([1 \quad 0], [K_d \quad K_p \quad K_i], T_s, \text{'tustin'}) \tag{2.12}$$

2.2. Literature on Asymmetrical Process Controllers

As discussed so far, PID controller is a generic method on tuning the processes. However, it may not be sufficient for all the systems especially if the system dynamics change in different modes as in the asymmetrical structures. Therefore, researchers work on this problem and add some new features on the standard PID controller. Following sections offer some new methods accordingly which apply different techniques in addition to PID controller.

2.2.1. Gain-Scheduled Control Method

This method was developed by the researches from the department of Electrical Engineering at National University of Singapore. They offer a new tuning method which uses a novel gain-matching and frequency-matching procedure to correct the asymmetrical oscillation phenomenon. The method adds a self-organising gain and a delay element to the basic relay method so that the two modes will have the same ultimate gain and ultimate

frequency. Finally, it is aiming to make the system oscillation behaving symmetrical. Here, the relay auto-tuning method was described in Section 2.1.2.3

The method first claims that the degree of the asymmetry should be determined whether applying gain-scheduling control would be worthwhile or not. Following is a proposal to get this information:

$$\delta = \frac{A^+}{A^+ + A^-} = \frac{T_h}{T} \tag{2.13}$$

where A^+ and A^- are the areas of the positive and the negative cycles of the relay output where:

- Asymmetry is low to moderate: $0.4 < \delta < 0.6$
- Asymmetry is high: otherwise

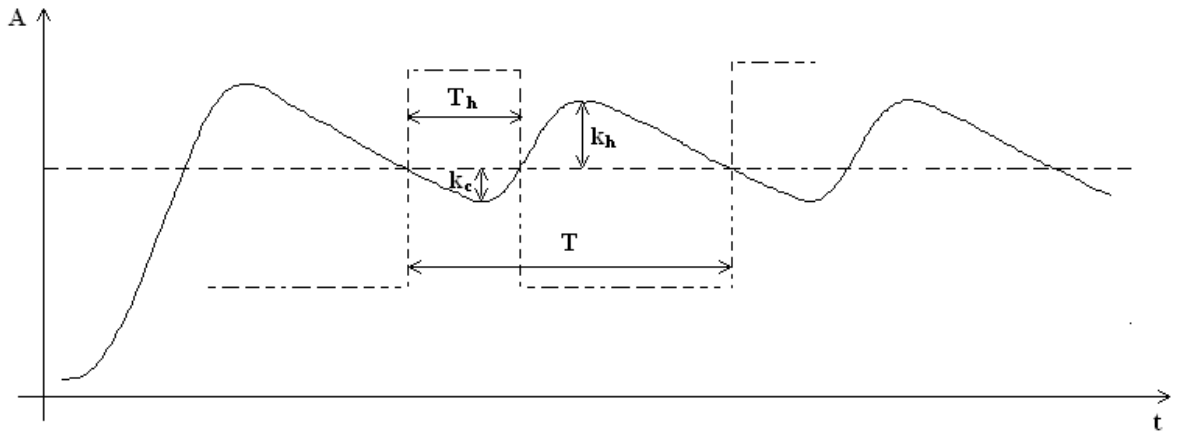

Figure 2.6. A typical asymmetrical cycle with control signal.

Then, the new function cascaded to the relay feedback system as shown in Figure 1.8 is as follows:

$$u(t) = f[v(t)] = \begin{cases} ku_m, t_i^+ < t < t_i^- + \theta \\ -u_m, otherwise \end{cases} \tag{2.14}$$

t_i^+ and t_i^- are the time instances corresponding to the positive and the negative switches of the relay, respectively and:

$$k = \frac{k_c}{k_h} \tag{2.15}$$

$$\theta = \theta_{-1} + \frac{1}{2}T - T_h \tag{2.16}$$

where θ_{-1} is the delay term used in the earlier cycle.

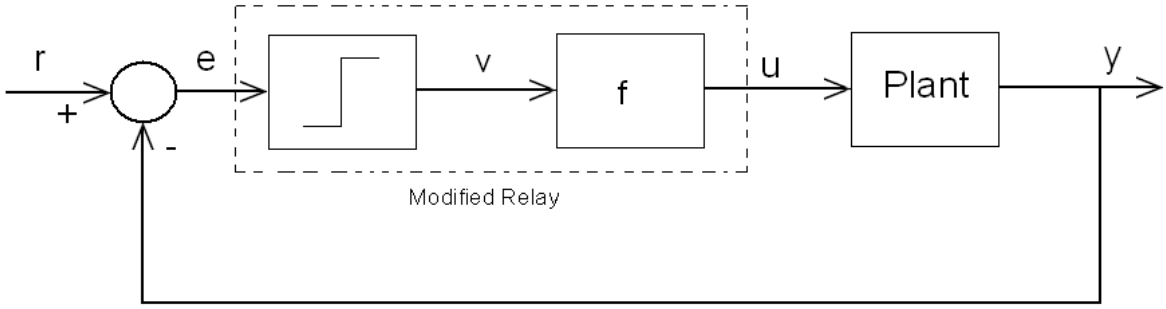

Figure 2.7. The modified relay feedback system.

Based on these models, two sets of PI control parameters are obtained accordingly, to satisfy the gain and phase margin reuirements of the system. Simulations using processes of various types of dynamics show the accuracy of this frequency response estimation, and the effectiveness of the approach for practical applications. [5]

2.2.2. Use of Biased – Relay Feedback

The researchers from Department of Chemical Engineering, National Taiwan Institute of Technology worked on the biased-relay feedback for system identification by taking care of the asymmetrical oscillations in feedback systems. They aim to get a relay feedback experiment that can identify two points on the Nyquist curve at frequencies $w = 0$ and $w = w_u$.

By the asymmetrical oscillations in a feedback system, they get two approaches for the steady-state gain (K_p).

For one, they considered the Dual-Input Describing Function (DIDF) which considers the input signal to the nonlinear element, e(t) consisting of two parts: a sinusoidal wave (a) and a biased signal (Δa). Finally, they get the resulting gain:δ_i

$$K_p = \frac{\Delta a}{\frac{2h}{\pi}\sin^{-1}\left(\frac{\delta_i - \Delta a}{a}\right)} \tag{2.17}$$

where δ_i is the bias in the relay.

Another approach to find K_p is applying a numerical method to get the following result:

$$K_p = -\frac{\int_0^{2\pi} u(t)dwt}{\int_0^{2\pi} e(t)dwt} \tag{2.18}$$

where u is the nonlinear element output.

For the system identification model, Luyben's (1987) original ATV method uses one relay feedback test for K_u and w_u determination. The method requires K_p to be known for the transfer function. Two test points are necessary for the aim, one is at w_u and another with phase angle around -130° and -140°. The proposed biased relay feedback requires only one point to get the process information at ultimate frequency and zero frequency.

After determining K_p, the researchers apply two model structure for system identification: first - and second-order plus dead time models. In model 1 (One Lag Plus Dead Time), once K_p is known then the deadtime, D can be found by:

$$\frac{1}{K_u} = \frac{K_p}{\sqrt{1+(\tau_p w_u)^2}} \tag{2.19}$$

$$-\pi = -\tan^{-1}(\tau_p w_u) - Dw_u \tag{2.20}$$

where τ_p is the time constant. If this calculated D is so different from the observed D, then Model 2 may be more useful.

In Model 2 (Two Unequal Lags Plus Dead Time), the deadtime is found over an observation then the following equations can be solved:

$$\frac{1}{K_u} = \frac{K_p}{\sqrt{\left[1+(w_u\tau_1)^2\right]\left[1+(w_u\tau_2)^2\right]}} \tag{2.21}$$

$$-\pi = w_u D + \tan^{-1}(-w_u\tau_1) + \tan^{-1}(-w_u\tau_2) \tag{2.22}$$

Based on the concept of dual-input describing function DIDF, the input-biased relay feedback experiments are proposed for the system identification. Results show that the biased relay feedback gives satisfactory performance under realistic proces environment. [6]

2.2.3. An Automatic Tuning Procedure for Unsymmetrical Processes

The model proposes a modification of the relay autotuner by fitting two low order process models to the relay feedback data needed in the gain schedule. method. The model assumes a well isolated furnace and describes the process in the heating phase as:

$$\begin{aligned} \dot{T_w} &= g_{heat}(u - T_w) + c(T_i - T_w) \\ \dot{T_i} &= c(T_w - T_i) \end{aligned} \tag{2.23}$$

Moreover, in the cooling phase as:

$$\begin{aligned} \dot{T_w} &= g_{cool}(u - T_w) + c(T_i - T_w) \\ \dot{T_i} &= c(T_w - T_i) \end{aligned} \tag{2.24}$$

where u is the control signal, T_i is the inside temperature, and T_w is the wall temperature of a furnace. On the other hand, g_{heat}, c and c_o are the heat transfer coefficients.

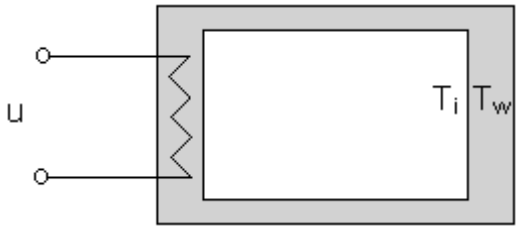

Figure 2.8. Schematic diagram of a furnace.

The steady-state representation of the furnace dynamics is as follows:

$$\dot{x} = \begin{pmatrix} -(g+c) & c \\ c & -c \end{pmatrix} x + \begin{pmatrix} g \\ 0 \end{pmatrix} u$$
$$y = \begin{pmatrix} 0 & 1 \end{pmatrix} x \tag{2.25}$$

The method aims to fit the model to the relay feedback data namely the parameters, c, g_{heat} and g_{cool}. The solution can be fitted to the state-space equation (2.18).

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_u}^{t} e^{A(t-s)}Bu(s)ds \tag{2.26}$$

The model determines two states x_1, x_2 and makes the following approximations:

t_a: $y = x_2$ crosses the upper hysteresis level, and u switches from u_{max} to u_{min}.
t_b: $y = x_2$ reaches its maximum and $x_1 = x_2$.
t_c: $y = x_2$ crosses the upper hysteresis level on its way down.
t_d: $y = x_2$ crosses the lower hysteresis level, and u switches from u_{min} to u_{max}.
t_e: $y = x_2$ crosses the upper hysteresis level again.

In time notations:

$$t_{ab} = \frac{t_a + t_b}{2}$$
$$t_{de} = \frac{t_d + t_e}{2} \tag{2.27}$$
$$t_{cd} = \frac{t_c + t_d}{2}$$

By the way, the method uses the following approximations:

$$\dot{x}_1(t_{ab}) \approx \frac{x_1(t_a) - x_1(t_b)}{t_a - t_b}$$
$$\dot{x}_1(t_{de}) \approx \frac{x_1(t_d) - x_1(t_e)}{t_d - t_e} \tag{2.28}$$

and finally,

$$\widetilde{c}_1 = -\frac{b}{2a} + \sqrt{\left(\frac{b}{2a}\right)^2 - \frac{q}{a}} \tag{2.29}$$

where a, b and q are resulted from some tedious calculations.

A modification of the relay autotuner that manages to obtain these two sets of controller parameters has been presented. A robustness analysis is also provided, which shows that the method is suitable even for processes that differ from the assumed model structure. [7]

2.2.4. Automatic Tuning of Nonlinear PID Controllers for Unsymmetrical Processes

The researchers worked on the autotuning of nonlinear PID controllers for unsymmetrical systems by extending the relay auto-tuning method. The approach is based on two estimation methods from a relay test. The methods are parameter estimation and pole estimation.

Parameter estimation method works works with the equations 2.16 and 2.17 to finalize them as:

$$\ddot{y}(t) + a\dot{y}(t) + by(t) = bu(t) \tag{2.30}$$

where t = T_i(t) and $a = 2c + g$, $b = gc$.

When we integrate it, we get the following result:

$$\dot{y}(t) - \dot{y}(t_0) + a(y(t) - y(t_0)) + b\int_{t_u}^{t} y(t)dt = b\int_{t_u}^{t} ud(t) \tag{2.31}$$

Then the cooling and the heating phase methods can be realized.

Pole estimation method considers the measurement of the output only without considering the output derivative since it is usually immeasurable in process industry.

Here, equation 2.23 has two real characteristic roots:

$$\lambda_{12} = -\frac{-(2c+g) \pm \sqrt{4c^2 + g^2}}{2} \tag{2.32}$$

By the inequality, $2\lambda\beta \le \lambda^2 + \beta^2$ and some tedious calculations,

$$\begin{aligned} t_b &= -\frac{1}{4\lambda_1} + t_a \\ t_c &= -\frac{1}{2\lambda_1} + t_a \\ t_d &= -\frac{1}{2\lambda_1} + t'_a \end{aligned} \tag{2.33}$$

where t_a, t_b, t_c and t_d are the time instants in two phases.

The proposal offers using two PID controllers each for one operating system by using the pole placement method to find the PID settings. The transfer function is described as:

$$G(s) = \frac{1}{K(\frac{s}{-\lambda_1} + 1)(\frac{s}{-\lambda_2} + 1)} \tag{2.34}$$

A PID controller and the settings are specified as:

$$G_c(s) = \frac{K(1 + sT_i + s^2 T_i T_d)}{sT_i} \tag{2.35}$$

$$K = \frac{w^2(1+2\xi)-\lambda_1\lambda_2)}{\lambda_1\lambda_2} \tag{2.36}$$

$$T_i = \frac{w^2(1+2\xi)-\lambda_1\lambda_2)}{w^3} \tag{2.37}$$

$$T_d = \frac{w(1+2\xi)-\lambda_1-\lambda_2}{w^2(1+2\xi)-\lambda_1\lambda_2} \tag{2.38}$$

The unsymmetrical parameters are chosen to be:

$$\xi_{heat} = 1.2, \xi_{cool} = 1.5, w_{heat} = w_{cool} = 5 \tag{2.39}$$

Additionally, symmetrical ones:

$$\xi_{heat} = \xi_{cool} = 1.5, w = 5 \tag{2.40}$$

On the other hand, the parameter estimation method gives:

$$C_{heating} = 0.359, g_{heating} = 0.6408, C_{cooling} = 0.5, g_{cooling} = 0.0644 \tag{2.41}$$

A relay auto-tuning of nonlinear PID controllers for unsymmetrical processes has been developed in this paper. Two new auto-tuning methods are developed. Especially, the pole estimation method only needs measurement wof the process output. [8]

3. REALIZATION OF ASYMMETRICAL SYSTEMS CONTROL

As we discussed in previous chapters, the asymmetrical systems like temperature controllers behave with different dynamics in two modes. This behaviour also depends on the temperature value at which the system relies on. For example, consider the Figures 3.1, 3.2, and 3.3 where an on-off control is applied around a set value. The system is heated upto the set value and then rested. As it is seen from the figures, the asymmetry is more severe at low temperature values. This means that we should work on a dynamic control system model to overcome the asymmetrical effects.

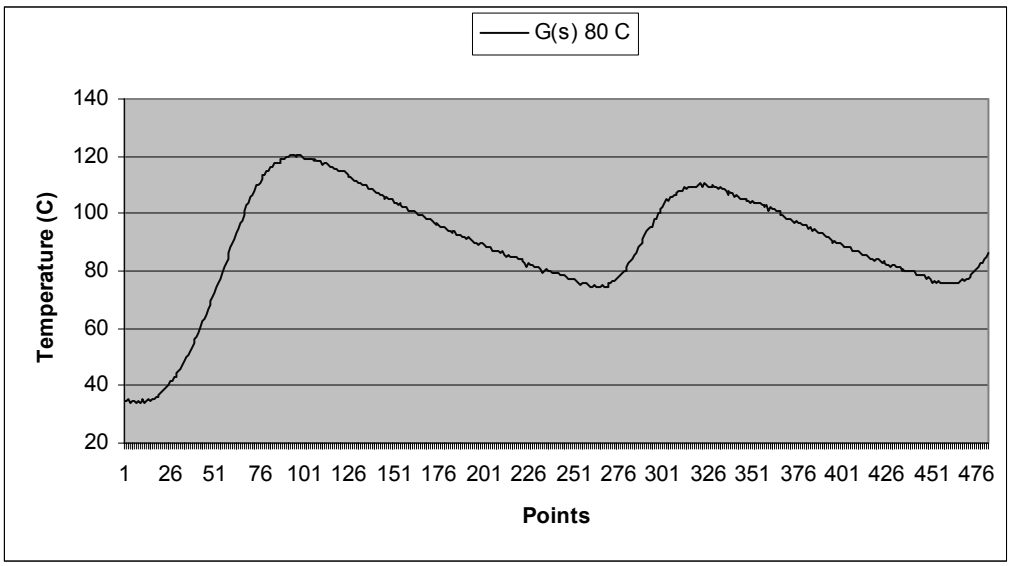

Figure 3.1. Asymmetrical behaviour of a 40 W Resistance at 80 °C.

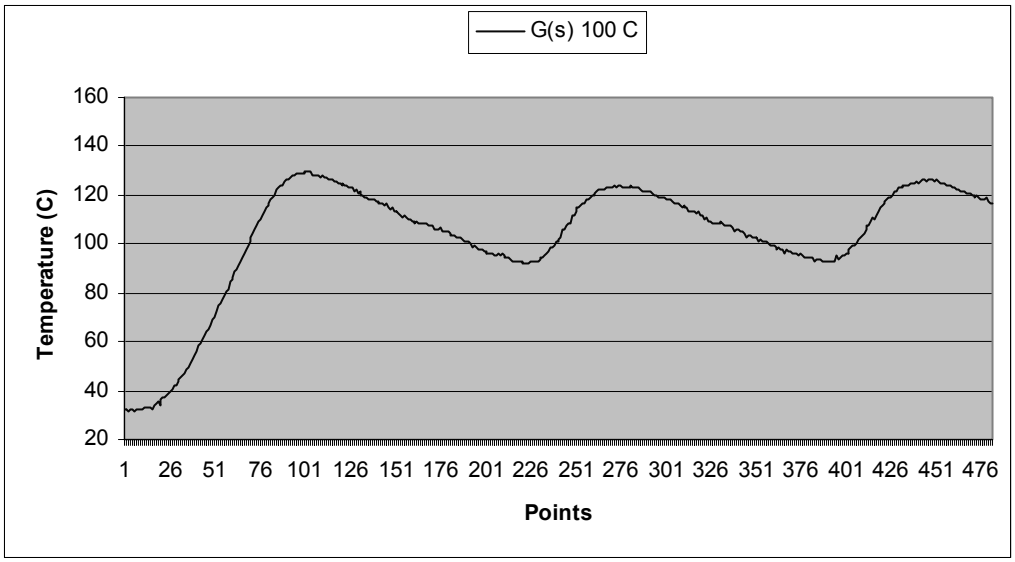

Figure 3.2. Asymmetrical behaviour of a 40 W Resistance at 100 °C.

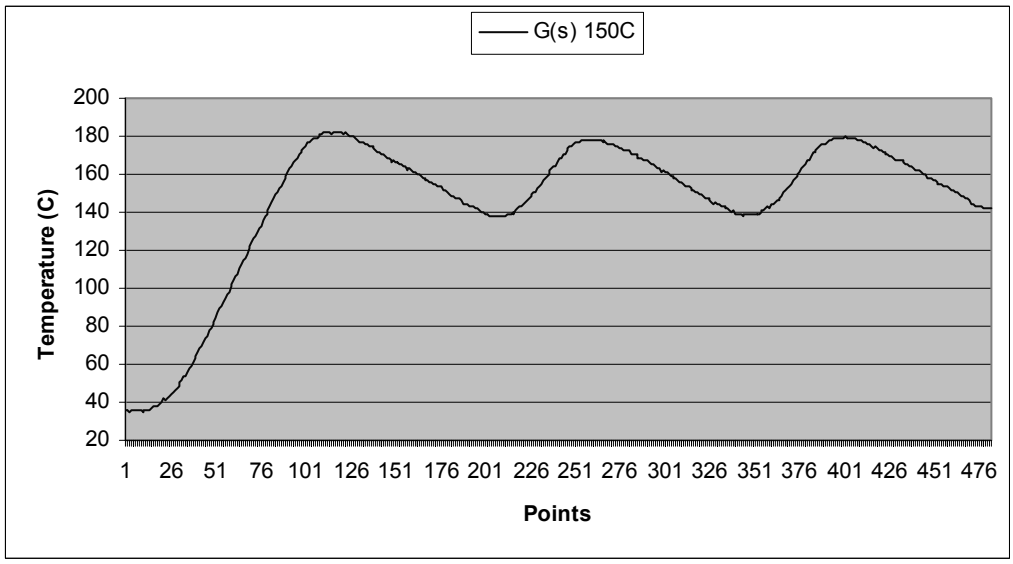

Figure 3.3. Asymmetrical behaviour of a 40 W Resistance at 150 °C.

In order to handle with those problems, we applied different modes so that each one improves different part of the whole control system. Therefore, in our study, we used four steps to realize the Asymmetrical Systems Control. These steps are Full Power, PWM, G(s) and PID Modes. Each step is followed by the next one. Following sections explain in details what those steps are and how to realize them.

3.1. Full Power Mode

Almost all temperature control systems have initial part to load PID with some amount and settle around the set temperature value. However, standard PID controller results in a long settling time and makes a high overshoot. This is because the difference between the initial and the final values loads PID with full performance and also importantly, due to the dead band characteristic of the materials, the system cannot easily cancel the oscillations during the first cycles.

The deadband is an area of a range where no action occurs at the output of the system. During the dead time, the system is loaded even no change occurs at the output and this is the reason why the controller cannot control the system output although applying a negative signal in time. In other words, the loaded system will maintain its flow for a while.

This is a common problem for almost all the processes and makes some difficulty for the controllers. If we apply a standard PID algorithm without considering this behaviour, we may get an undesired system output as shown in Figure 3.4.

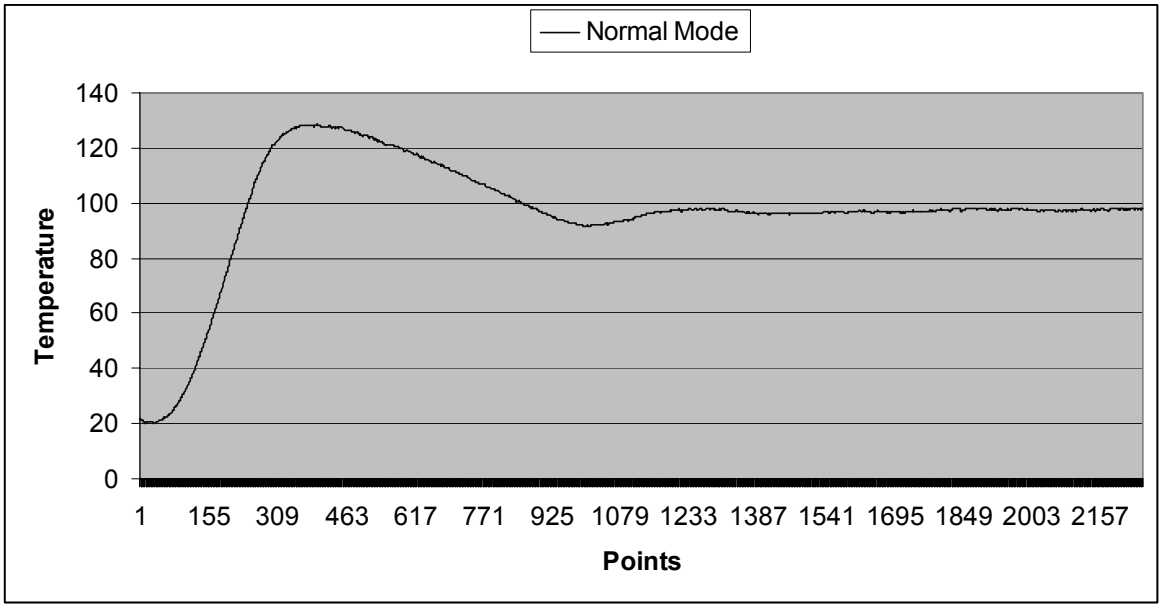

Figure 3.4. System Output during Normal Mode for 40 W Resistance

Full Power mode was developed to overcome these problems. It is the initial part of the full control system and speeds up the system while minimizing the overshoot as shown in Figure 3.5. Actually, Full Power means applying 100 per cent PWM upto a pre-determined value. By the way, the aim is to make the system output approach to the setpoint.

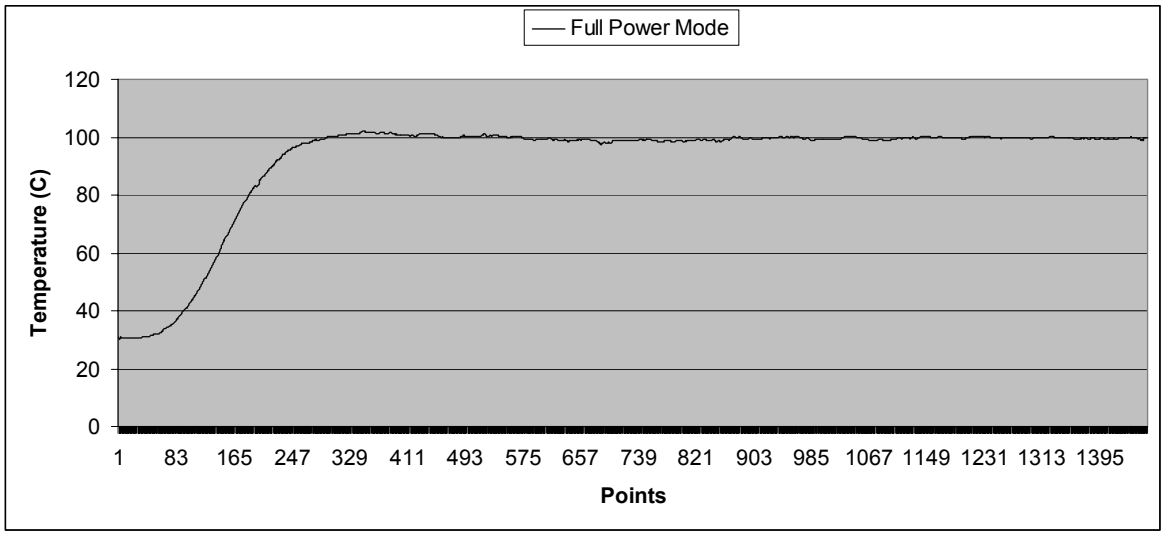

Figure 3.5. System Output with the effect of Full Power Mode for 40W Resistance at 100 °C

In this mode, the system is tuned according to the set values. Then, for the desired set value, the corresponding full power point is set and the system is initialised by full power upto this point.

Figure 3.6 shows different Full Power Mode outputs for different set points of a 40 W Resistance. During the experiment, we have applied four test points to get some information about the system. The corresponding experiment results are shown in Table 3.1. As it is seen that the increase in the Full Power set temperature is almost proportional to the increase in the final temperature. For example, while the change from 50 °C to 75 °C (50 per cent) in set temperature results 23,81 °C change in the final temperature, the change from 50 °C to 100 °C (100 per cent) results 45,05 °C change. That means the ratio of the set temperatures is proportional to the ratio of the changes in the final temperatures. Therefore, only a few good tuning points are enough for Full Power Tuning of the full system.

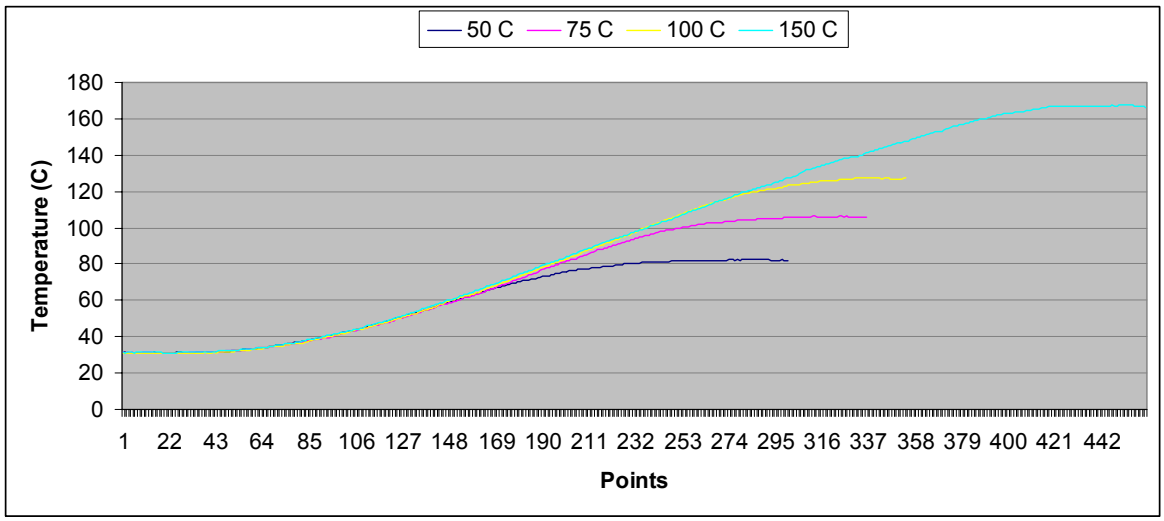

Figure 3.6. Full Power Mode Outputs for different set points for 40 W Resistance

Table 3.1. Maximum temperature values for Figure 3.6.

Set temperature (°C)	50	75	100	150
Final temperature (°C)	82,14	105,95	127,19	167,66

3.2. PWM Mode

PWM is the mode applied between the Full Power and the PID Modes. In this mode, a constant PWM signal is applied as the input to the system. The PWM value is pre-tuned for different set values.

This mode is applied to stabilize the system around the set point and support the Full Power Mode.

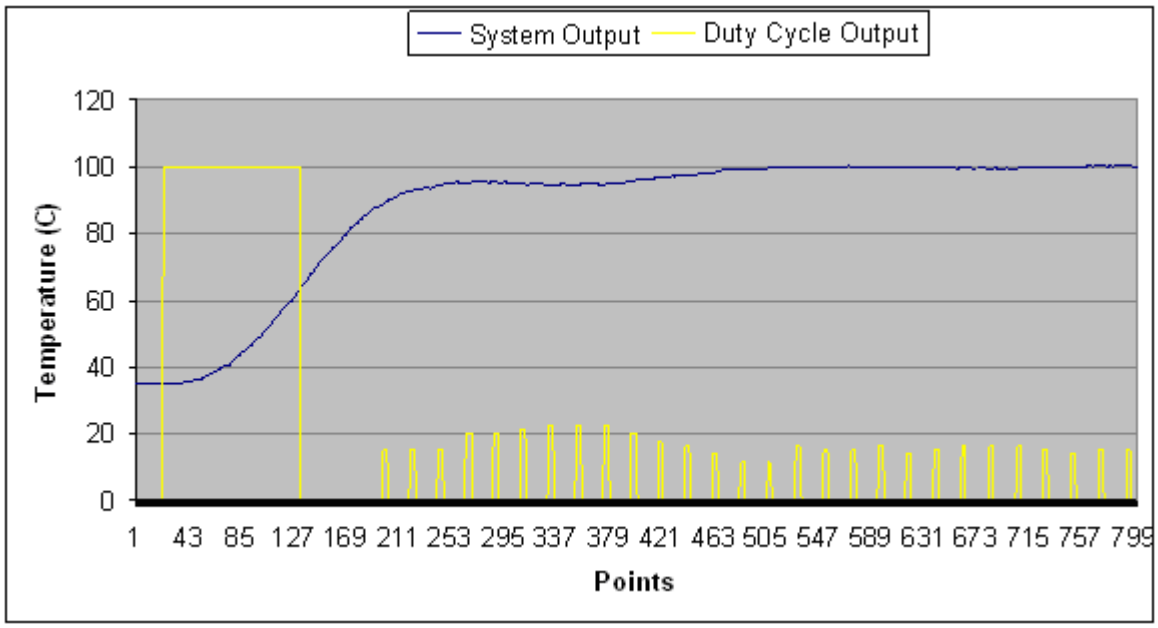

Figure 3.7. System and Duty Cycle Outputs with Full Power and PID but without PWM Modes.

In Figure 3.7, we have applied only two modes; Full Power and PID but not PWM. As it is seen on the graph, the temperature increases sharply with Full Power mode upto a near to the setpoint. However, when the transition to the PID mode, the output switches to a stable and then a negative state for a while. This is because there is no power applied to the system between the points 127 and 211. Within this region the system gets back into the deadband state again and waits for a while so that PID mode loads the system again.

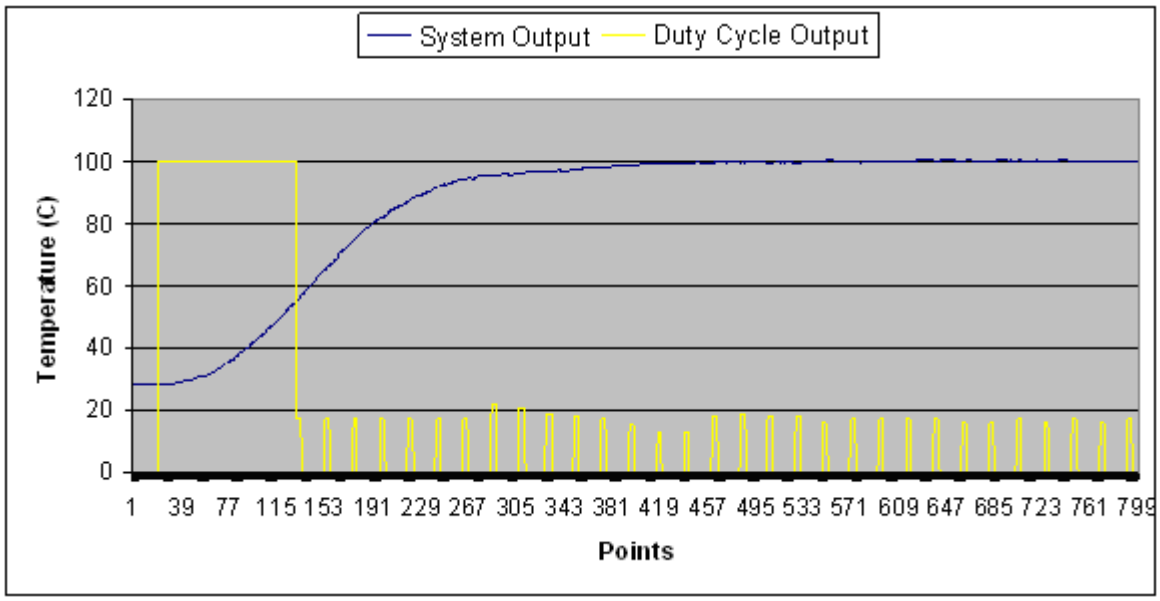

Figure 3.8. System and Duty Cycle Outputs during the Full Power, PWM and PID Modes together for 40 W Resistance

On the other hand, we have added a PWM mode into the system and repeated the experiment again. Figure 3.8 shows the system output during the Full Power, PWM and PID Modes together. The experiment is realized with the given parameters in Table 3.2. PWM Mode starts just after Full Power Mode completed. Actually, PWM signal binds on the Full Power signal. By the way, the signal is approached to the set point smoothly.

As it is shown now that PWM mode improves the problem in Figure 3.7 during the transition to PID mode. It both cancels the deadband effect and decreases the settling time.

Table 3.2. Parameters used in Figure 3.8.

Set Temperature (°C)	Full Power Temperature (°C)	PWM ratio (per cent)
100	52	15

Figure 3.9 shows a PWM Mode output for 15 per cent duty cycle for a 40 W Resistance. Increasing the number of tuning points will increase the tuning time but on the other hand, it will improve the performance of the whole system.

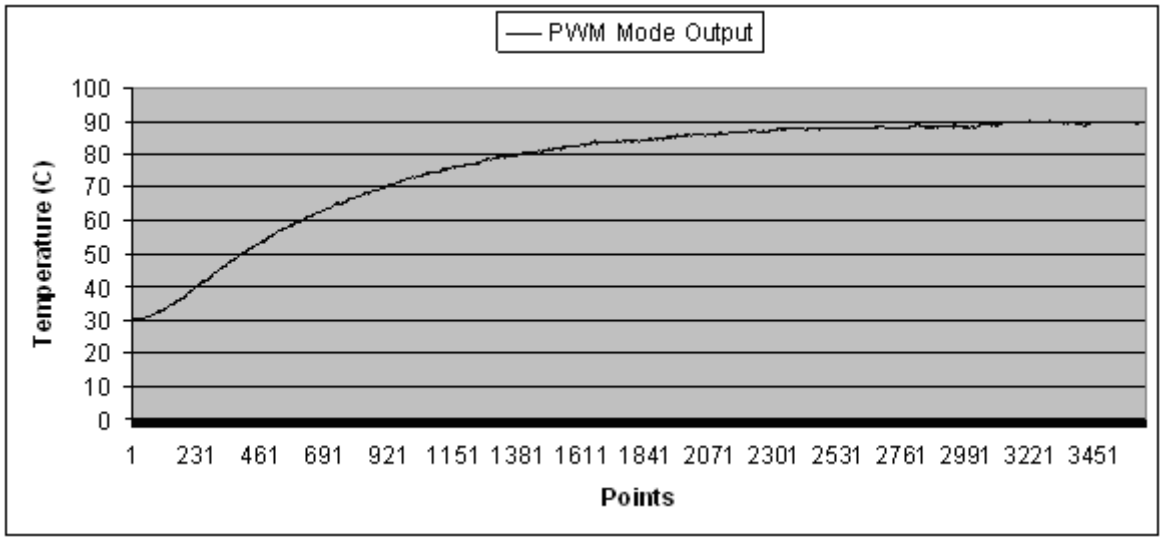

Figure 3.9. PWM Mode Output for 15 per cent Duty Cycle of 40 W Resistance

3.3. G(s) Mode

One of the important parts of the control system design is to derive the PID parameters (namely K_p, K_i, and K_d). If we approximate the transfer function (G(s)) of the system, we can obtain the controller parameters as follows:

Basically, we can assume the transfer function in Equation 2.5 and apply some known equations to solve the problem. This is the normal and simple procedure but we should find G(s) at this point.

Consider the Relay Method again in Section 2.1.2.3. All the details to derive the transfer function are explained here.

Table 3.3 shows the input parameters to apply Relay Method for our system. By the table, the method will be applied with 100 °C set temperature. While the total duration for one cycle is 6.5 seconds, the system will be refreshed with 3.3 Hz frequency.

Table 3.3. Input parameters to apply Relay Method.

Set Temperature(°C)	Hysteresis (per cent)	Period (s)	Sampling Frequency (Hz)
100	5	6.5	3.3

First of all, the system must get into limit cycle so that it will have repeated cycles. After making many experiments, we recognized that limit cycle occurs in the second cycle. So, we should wait during the first cycle and consider the next ones starting from the second cycle. For the simplicity, we take the two cycles within the limit cycle. Figure 3.10 shows the system output during the Relay Method.

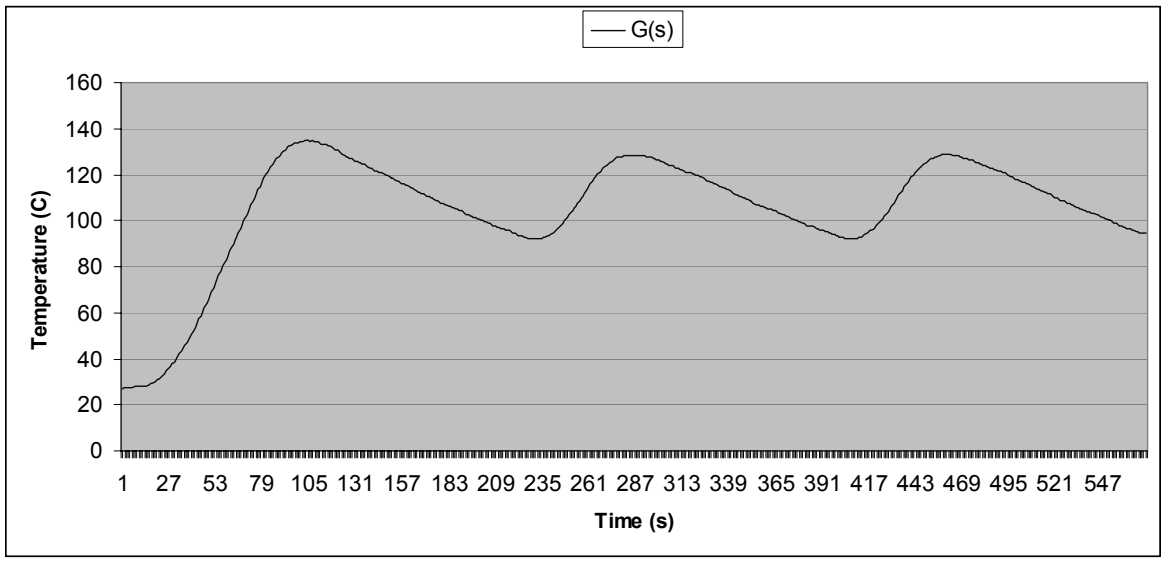

Figure 3.10. Relay Method response of 40 W Resistance for 100 °C

The Relay method parameters are shown in Table 3.4. Here, a is the amplitude of the signal, d is the relay output and T_c is the ultimate period.

Table 3.4. Relay Method parameters for Figure 3.5.

a	D	T_c
29	700	180

PID parameters can be calculated by the equations given in Table 2.1. The results can be seen on the first row of Table 3.5.

Table 3.5. Calculated PID parameters for Figure 3.5.

K_p	**K_i**	**K_d**
18.45	0.0111	21.6

The parameters' values seem to be logical and the PID range can be adjusted according to them so that the parameters can work effectively. The parameter values for different temperatures can be seen on Table 3.6.

In G(s) mode, the Variable K_i is also calculated but this is described in Variable K_i section, 5.4.3.

3.4. PID Mode

The modes so far are for the initial part of the whole control system. They are to decrease the settling time, decrease the initial overshoot and smooth the signal around the set temperature.

After these modes are completed, the control system has to succeed to settle the system at the desired point and then never allows it to make any overshoot or undershoot. This is the major issue of all the control systems but at the same time the most difficult part.

Standard PID algorithm is not sufficient itself to succeed this purpose. It becomes a more serious problem in our case since we are dealing with the asymmeytric systems. However, hopefully, by making some changes on the standard algorithm and adding some new features, the system can behave like a symmetrical one and by the way, it can succeed in reaching the desired outputs. We will apply some techniques in the following sections to achive the criteria.

Also note that PID Mode starts as soon as the current temperature exceeds 95 per cent of the set temperature after PWM Mode.

3.4.1. Changing PID Values

Tuning a PID control system is a preliminary part of the whole system. It is to get some information about the system so that we learn how to determine the input parameters. Generally, the tuning is done based on a specific set value. However, one point may not be sufficient for all the systems especially if your system has a wide range of working area.

In our system, since we aim to realize a general control application for the asymmetrical systems, it needs to be able to work in a large temperature interval. Therefore, the system requires more than one point tuning procedure.

For this purpose, we choose different set values within the working range by some equal intervals between them. Then, we apply PID tuning fo each set value.
During the practice, the pre-tuned set points are compared according to the current set value and the tuning parameters belonging to the point which is the best and the nearest one to the current set value is chosen and applied. By this way, the tuning parameters are selected for a better performance.

This method is more suitable especially if there are more than one set value during the application. In this case, for each changing set value, the pre-tuned parameters are automatically chosen according to that set value and system continues working with the new PID values.

Table 3.6 shows the PID tuning parameters for different set temperatures.

Table 3.6. PID tuning parameters for different set temperatures for 40W Resistance

Temperature (°C)	**K_p**	**K_i**	**K_d**
80	17.05	0.01	23.88
100	18.45	0.011	21.6
150	18.64	0.014	17.04

3.4.2. Set Integral Term

In the initial state, total PID is zero. However, we are not in this state now, so PID should not be zero. It has to be loaded by some value to drag the temperature to the set value. Before PID Mode, we were applying PWM Mode and it was the constant PWM input to the system. Now, PWM ratio will be changeable by PID and the previous constant PWM ratio would be a good starting point for this mode. That is, we will load PID with some value so that it will result the previous PWM Mode ratio and then take its role. Here, since K_i parameter will be dominant while approaching to the set value and PID will be lead by that, loading the integral term would be the ideal way to drive the system in PID Mode.

Basically, the equation would be:

$$Integral = \mathrm{K}_i \sum_{k=0}^{n} \mathrm{e(k)} = (PWM\,\%) \times PID_{\max} - differential - proportional \qquad (3.1)$$

However, the problem here is that while the system is approaches to the set value, due to the decrease in the proportional term, PID decreases, as well. Therefore, the system will lose its power to drive through the set point and it will slowly increase the temperature. After making many experiments to handle with this problem, we concluded to apply PWM tuning by the consideration of 10 per cent above the set values. This change would decrease the settling time and get a sufficient PID loading around the set value.

The effect of setting the integral term can be seen in Figure 3.11. It is obvious that this feature fires the system again to go through the set value.

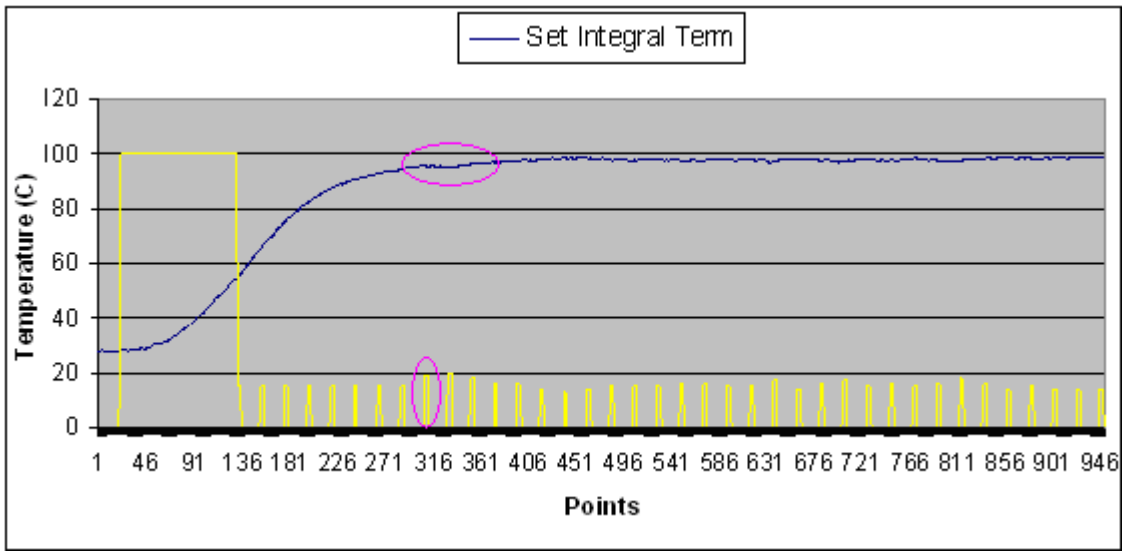

Figure 3.11. Effect of setting the integral term in PID Mode with the corresponding Duty Cycle graph.

3.4.3. Variable K_i

Assume Figure 3.10 to get some idea on the asymmetrric system behaviour. As it is expected, heating speed is higher than the cooling speed. That's why, it takes much time during the landing of the system to get the set point back.

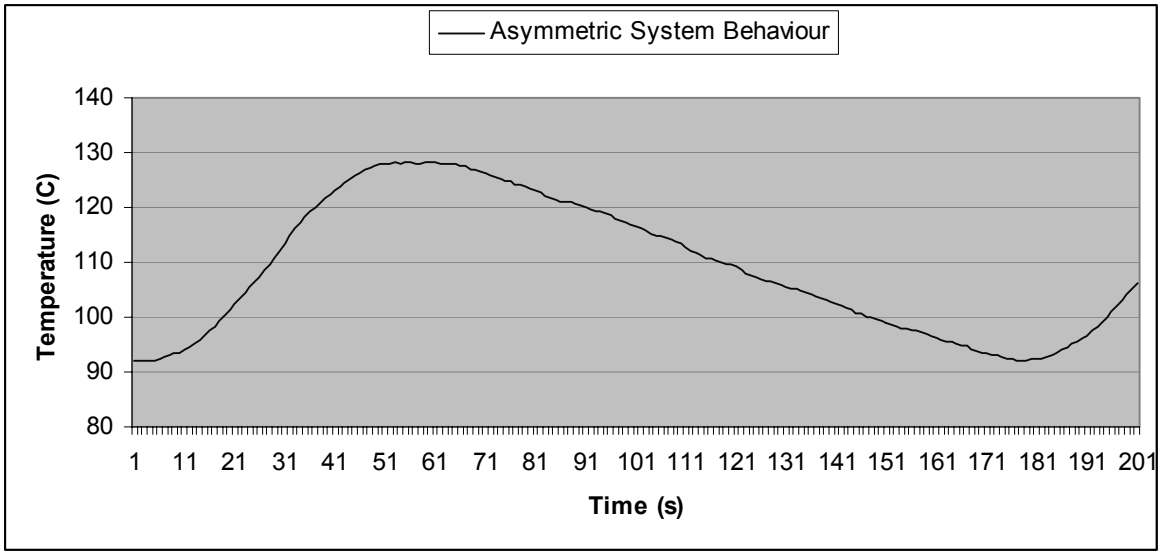

Figure 3.12. Asymmetrical system behaviour during one complete period for 40W Resistance at 100 °C

Now consider Figure 3.12 to get into more information about the structure. We get some following results about the asymmetrical system:

Table 3.7. Time taken during the rising and the landing for 40W Resistance at 100 °C

Rise time (s)	**Land time (s)**
195-147 = 48	147-19 = 128

Table 3.8. Total change in PID during the rising and the landing based on a constant K_i value for 40W Resistance at 100 °C

	Total change in PID during rising	**Total change in PID during landing**
Actual Value with *constant K_i*	3594,18	6504,75
Ratio(percent) with *constant K_i*	5,48	9,93

Table 3.7 and Table 3.8 give the difference in time and PID which are what we are looking for. In the calculation, we used a 16-bits resolution for total PID load which corresponds 0 – 65535 value interval and the ratios are based on this criteria.

It is a very important result that the change in PID during rising is relatively lower than the change during landing with the constant K_i value (almost half of that). This is due to the difference between the rising and the landing time as shown in Table 3.7. This is what the asymmetrical structure brings with the system.

Consider the equation 2.5 in terms of the error effect. Here, proportional and integral terms are directly related with the error value. That means higher the error higher the proportional and integral terms and so PID.

Now, consider the case in which the system is around the setpoint. In this situation, the error becomes so small and almost cancels the proportional term since:

$$P = \mathrm{K}_p \mathrm{e(n)} = 0 \text{ where } \mathrm{e(n)} \to 0$$

As we noted before, the derivative term has only small effect on the temperature control systems and it even almost loses this effect around the setpoint, then we can accept it as a non-effective term, as well.

$$D = \mathrm{K}_d (\mathrm{y(n)} - \mathrm{y(n-1)}) = 0 \text{ where } \Delta \mathrm{y} \to 0$$

Then the integral term becomes the dominant term and the system can be compensated by that. Since we aim to balance the system, then we can apply a variable K_i for the integral term.

$$I = k\mathrm{K}_i \sum_{\mathrm{k}=0}^{\mathrm{n}} \mathrm{e(k)} \tag{3.2}$$

where ‘k’ is the compensation constant to compensate the asymmetrical system over K_i.

In short, since cooling takes much longer time than the heating, PID change is much more in cooling state and when it gets into undershoot state, then it needs higher PID value to get the set point back. That corresponds applying different K_i values for the system and so we prefer K_i bigger during the undershoot.
Then parameters in equation 2.9 becomes:

$$a_0 = K_p + kK_i \frac{T_s}{2} + K_d \frac{1}{T_s}$$

$$a_1 = -K_p + kK_i \frac{T_s}{2} + K_d \frac{-2}{T_s} \qquad (3.3)$$

3.4.3.1. Calculation of K_i . Consider Figure 3.12 again. In this case we only focus on the first part of the period as shown in Figure 3.13.

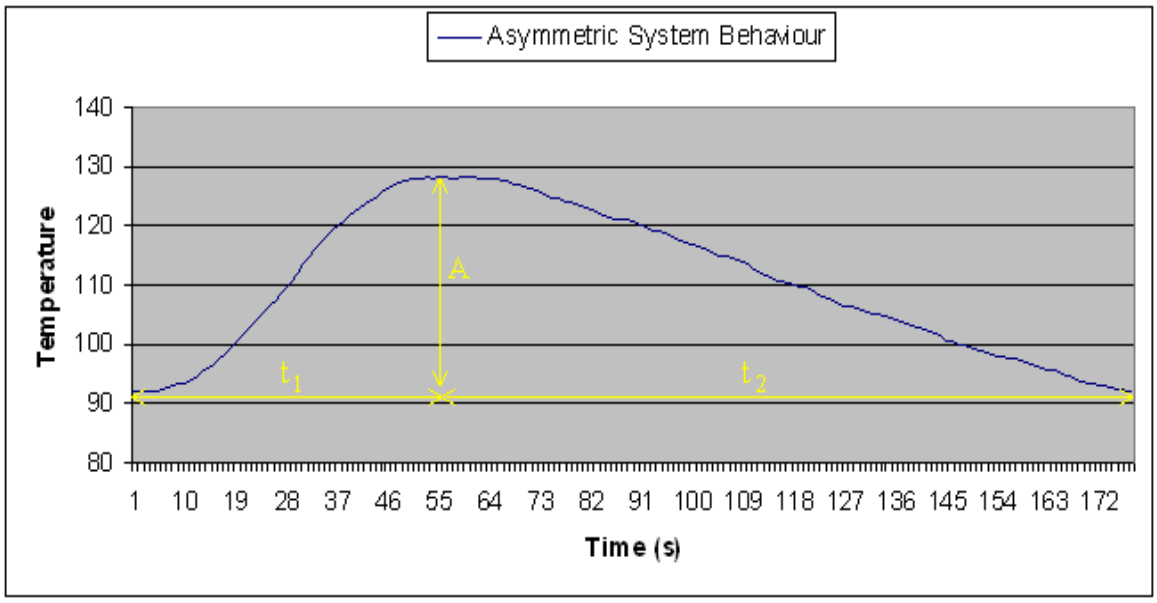

Figure 3.13. Asymmetrical system behaviour during first part of a complete period for 40W Resistance at 100 °C

If we divide the figure into the rising and the landing parts then it can be obviously seen that rising time (t_1) is less than the landing time (t_2) as expected. We can also recognize that the initial and the final temprature values for the complete figure are the same. Finally, when we aim to compare two parts, it is the best and the easiest way to work on the slopes of both parts. That is, the value of 'k' is the slope ratio. Actually, since the amplitude (A) is the same due to having the same initial and the final values, then the slope ratio is directly proportional to the time ratio of the parts. Finally, we can find the value of 'k' over the time ratio.

$$k = \frac{m_1}{m_2} \text{ where } m_1 = \frac{A}{t_1} \text{ and } m_2 = \frac{A}{t_2}.$$

Finally, we get:

$$k = \frac{t_2}{t_1} \tag{3.4}$$

After switching to apply the variable K_i on the system, we repeat the experiment and get the results into Table 3.9 by comparing with Table 3.8.

Table 3.9. Total change in PID during the rising and the landing based on a variable K_i value for 40W Resistance at 100 °C

	Total change in PID during rising	**Total change in PID during landing**
Actual Value with *variable K_i*	6757,06	6504,75
Ratio(percent) with *variable K_i*	10,31	9,93

The results are so optimistic since the changes in PID during the rising and the landing parts are now almost equalized as shown in Table 3.9. Here, the K_i values are retrieved from equation 5.2 and by the way the system is made behaving like symmetrical around the setpoint.

3.4.3.2. Application Results. There are some improvements on the control system with the variable K_i. In Figure 3.14, we used the standard PID with a constant K_i ($K_i = 1$) during the experiment and got the response. As it is shown, the system cannot settle the setpoint easily due to the asymmetrical structure. This is one of the main problems while getting some undershoot oscillations untill the setpoint is the another one.

When we increase K_i value to another suitable value in Figure 3.16 with $K_i = 5$, then we have different problems. The two main problems are getting a huge overshoot and an increase in undershoot.

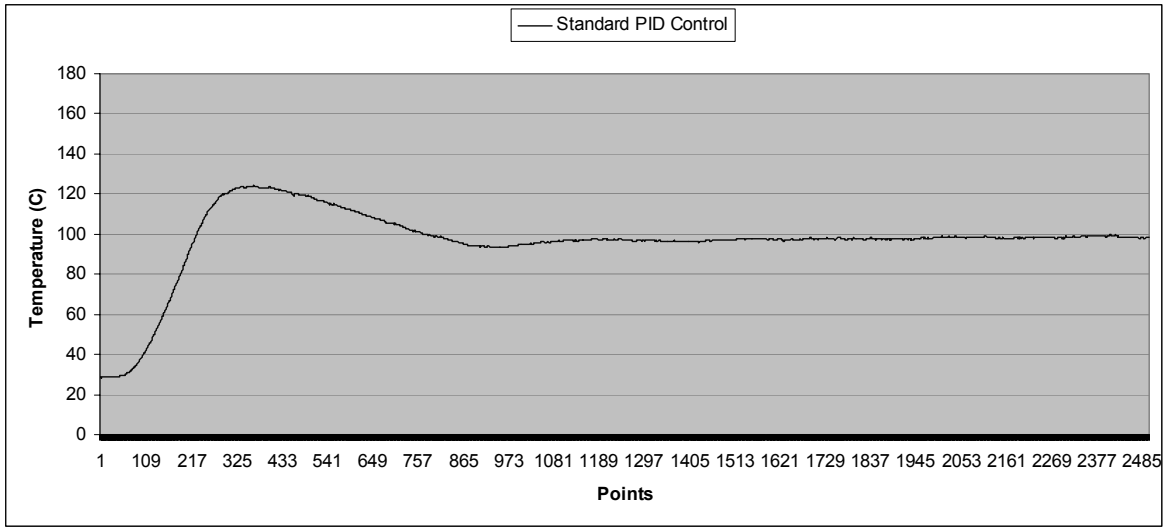

Figure 3.14. PID Control output for 40 W Resistance at 100 °C where K_i =1

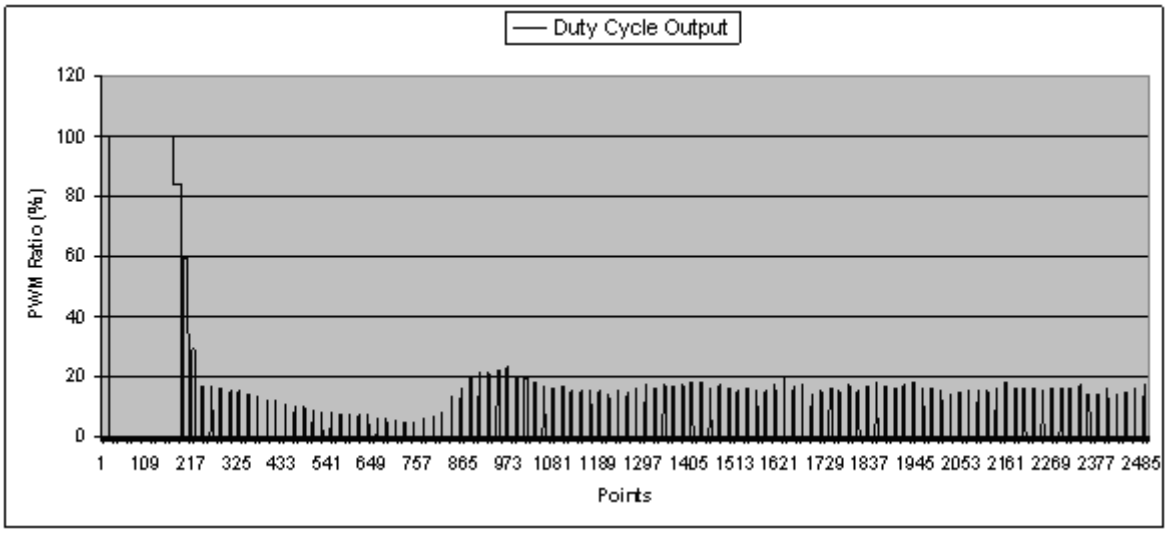

Figure 3.15. Duty Cycle Output of the system shown in Figure 3.13.

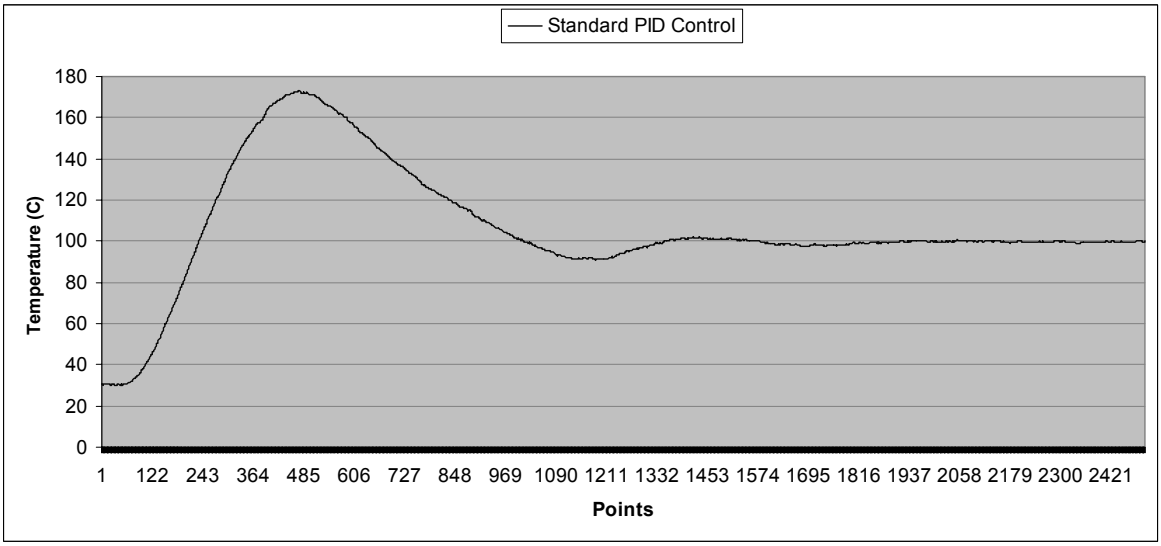

Figure 3.16. PID Control output for 40 W Resistance at 100 °C where K_i =5.

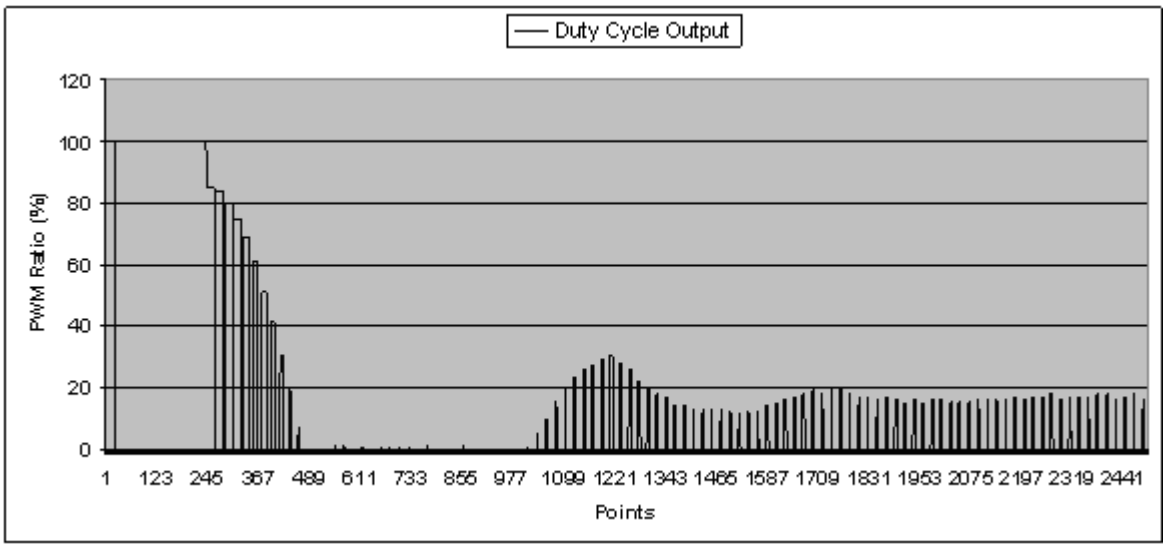

Figure 3.17. Duty Cycle Output of the system shown in Figure 3.15.

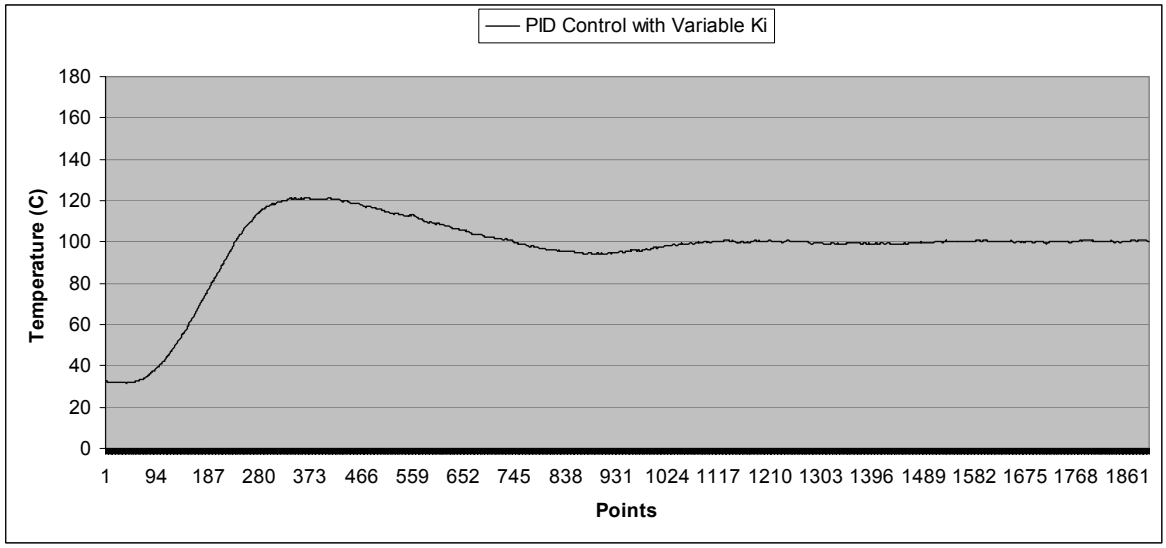

Figure 3.18. PID with Variable K_i Control output for 40 W Resistance at 100 °C where K_{i1} =1 and K_{i2} = 5

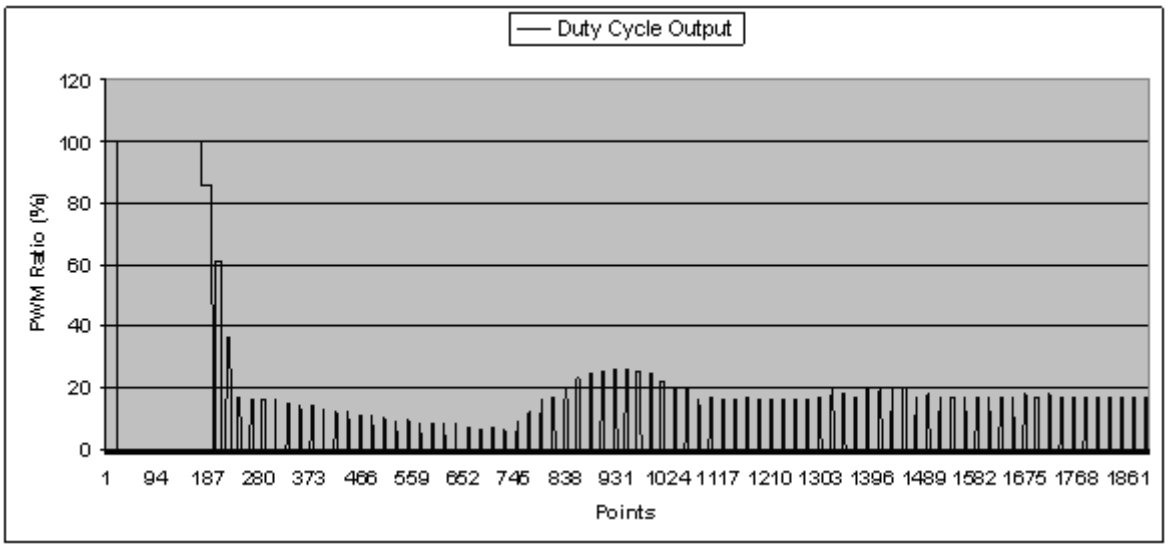

Figure 3.19. Duty Cycle Output of the system shown in Figure 3.17.

The problems here are caused of much loading the PID. This is because, the integral term gets loaded with a bigger value since K_i is greater now and the change in integral term will take some time to decrease.

As it can be seen from equation 2.5, there is no significant change for P and D terms. However, integral term (I) is stil in action as the effective term for PID due to the change in K_i.

Another problem with Figure 3.16 is its having some oscillations due to both the asymmetrical structure and the uncontrollable K_i value.

The problems with constant K_i are:

- Overshoot
- Undershoot
- Long settling time
- Steady-state error

Lastly, we apply a variable K_i as our new suggestion. As discussed in this section, the variable K_i improves the system response significantly as shown in Figure 3.18.

Variable K_i produces a lower overshoot and undershoot while reducing the settling time and reducing the oscillations. These improvements are the expected ones from an asymmetrical system to behave like a symmetrical one.

Table 3.10 gives the experimental results for three cases. The system reponse is improved with the variable K_i.

Table 3.10. Comparison of the systems regarding the standard PID with different K_i values and the PID with the Variable K_i for 40W Resistance at 100 °C

	Maximum Temperature	**Overshoot (per cent)**	**Minimum Temperature**	**Undershoot (per cent)**	**Settling Time(s)**
PID with K_i = 0.01	124.13	24.13	93.43	6.57	810
PID with K_i = 0.021	173.08	73.08	91.14	8.86	576
PID with Variable K_i	121.13	21.13	94.00	6.00	329

In Table 3.11, variable K_i experiment results are shown. From the data in the table, we can say that the asymmetrical behaviour of the system is more dominant at lower temperature values than the higher ones. For example, variable K_i is greater at 80 °C since the rising and the landing times take much different lenghts than any other temperature value higher than that. This behaviour affects the variable K_i efficiency during the applications.

Table 3.11. Variable K_i experiment results for different set values of 40W Resistance

	Rising Time, t_1 (s)	**Landing Time, t_2 (s)**	**Variable K_i (t_2/ t_1)**
PID Control at 80 °C	64	135	2,1
PID Control at 100 °C	60	113	1,88
PID Control at 150 °C	54	78	1,44

4. EXPERIMENTAL SYSTEM

4.1. Hardware Design

4.1.1 General Features

The hardware design module is to realize a control product which is applicable for the industry. The system is controlled by a microcontroller. The power supply unit supplies all the necessary powers to the system devices. The temperature of the heater is read over a thermocouple and linearized by the AD594 to be given to the microcontroller as the input. Then, the information is sent to the PC over ST-232 device. After all the calculations in PC, the PWM information is received and the microcontroller drives the relay according to PWM ratio to control the temperature of the heater. Meanwhile, the LCD displays the desired data on the screen.

Figure 4.1 shows the system block diagram according to these information.

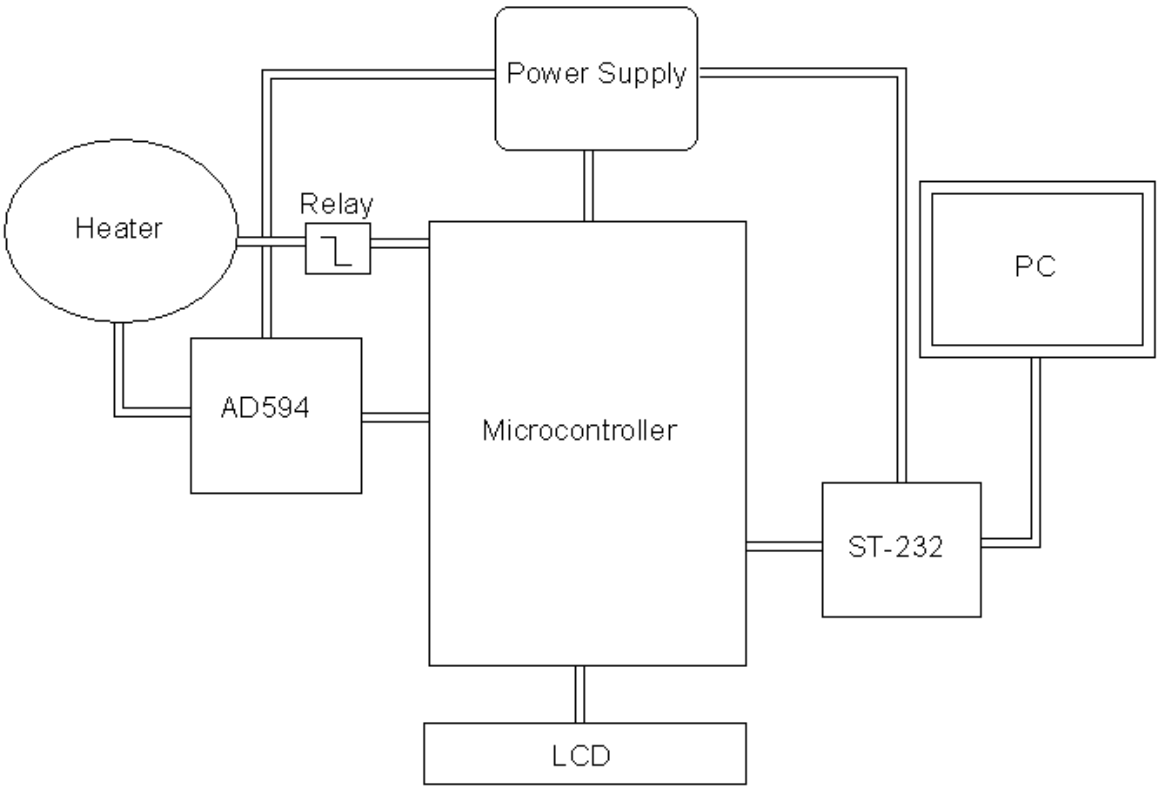

Figure 4.1. Block diagram of the Hardware module.

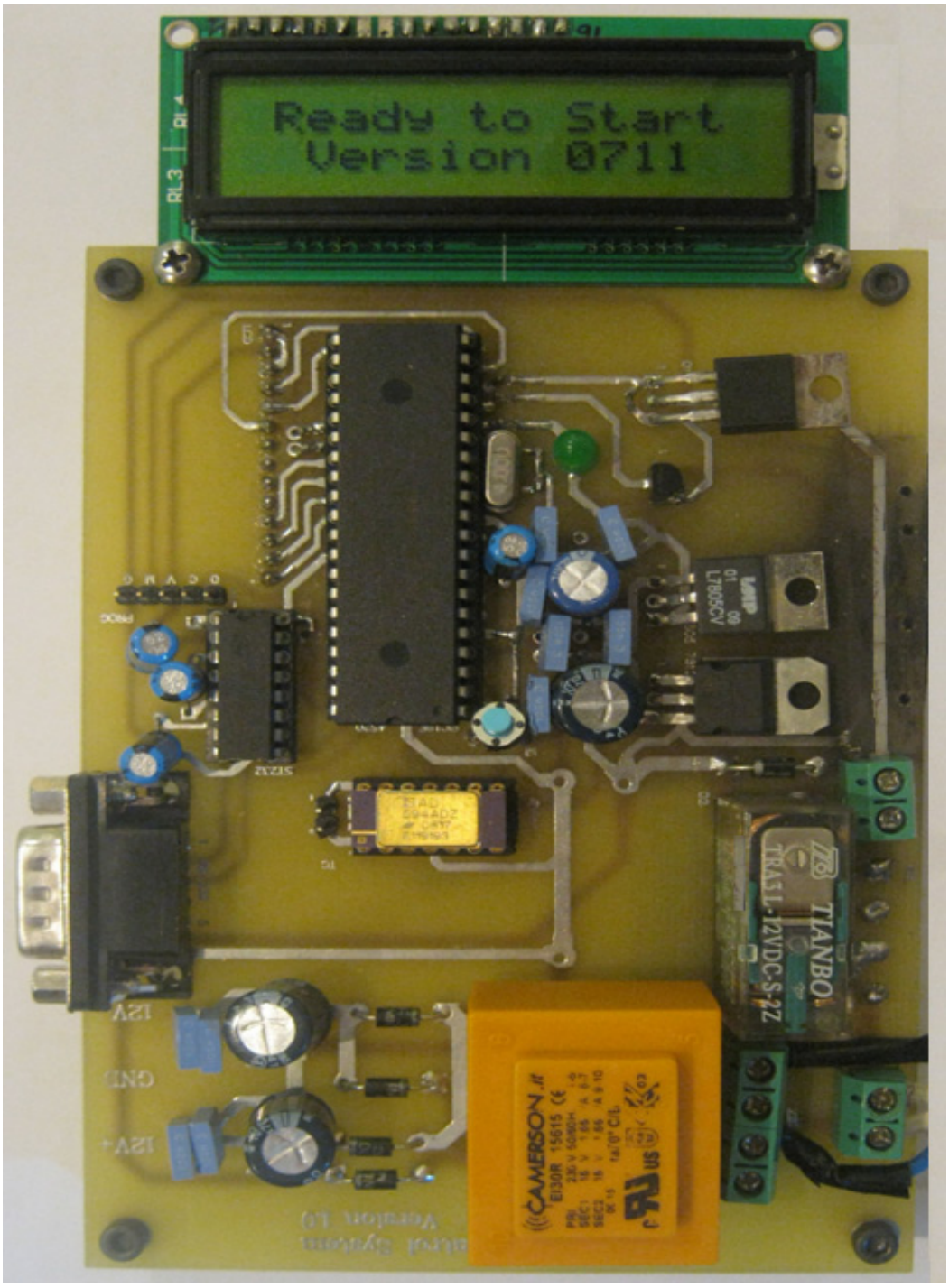

Figure 4.2. Hardware module of the control system

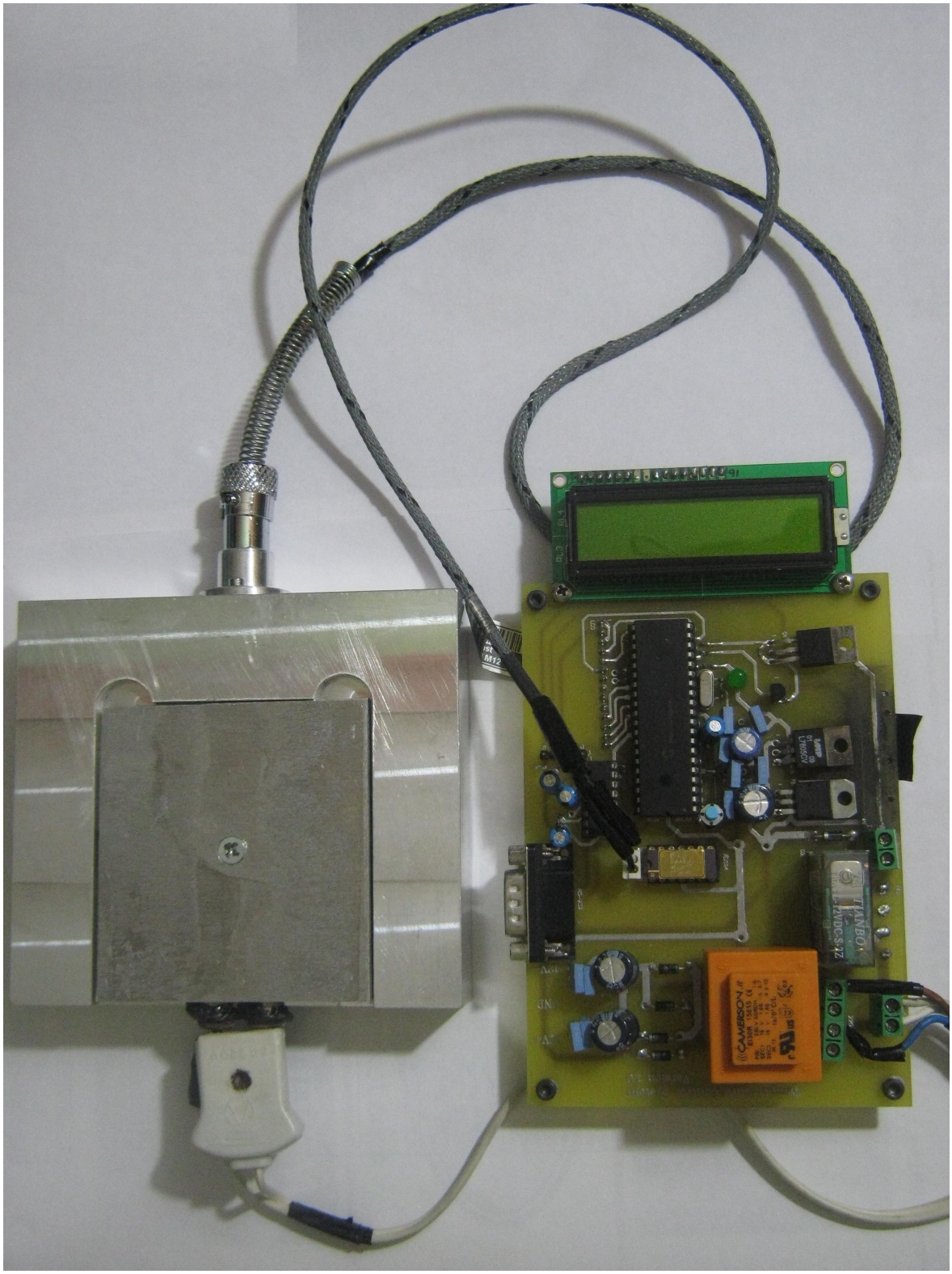

Figure 4.3. Experimental system

4.1.2 Power Supply Unit

The system uses a transformer device as the power supply unit. The specifications are shown in Table 4.1.

Table 4.1. Specifications for power supply unit

Input Voltage	Frequency	Output Voltage	Output Current	Total Power
220 V AC	50 / 60 Hz	12-15 V DC	3.32 A	50 W

Figure 4.4 shows the power supply circuit with the rectified output. The output is then given to the switch regulator as the input to be converted into the desired level as discussed in Section 4.1.3.

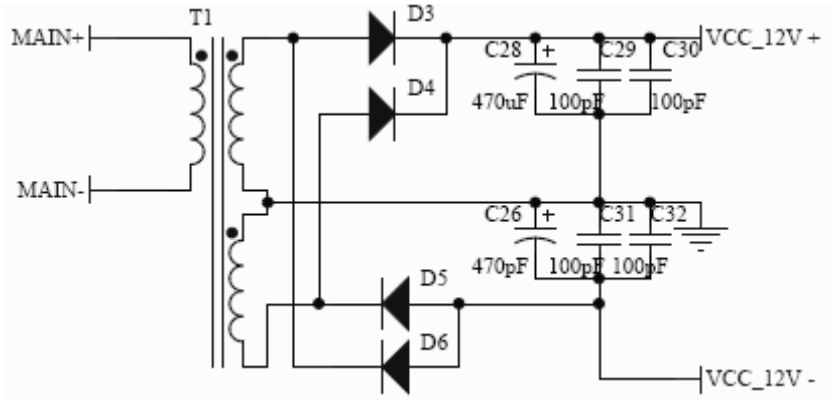

Figure 4.4. Power supply unit of the system

4.1.3 Switch Regulators

In our system, we need two voltage sources. Each voltage supply requirement and the applications are shown in Table 4.2.

Table 4.2. Voltage supply requirements of the system and the corresponding applications.

Voltage source	Applications
5 V	PIC18F4520, AD594, ST-232
12 V	Relay

There are two switching regulators for the sources. While L7805 is supplying 5V, 1A, L7812 supplies 12V, 1.5A power.

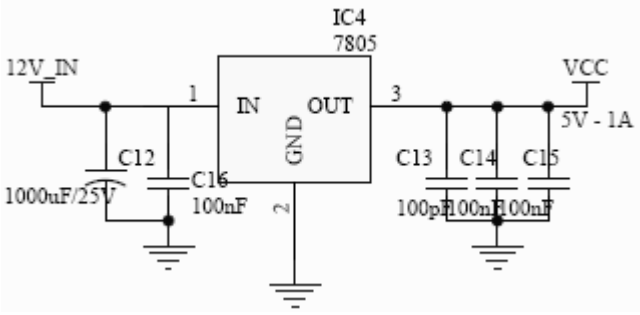

Figure 4.5. Switch regulator circuit for 5V, 1A power source.

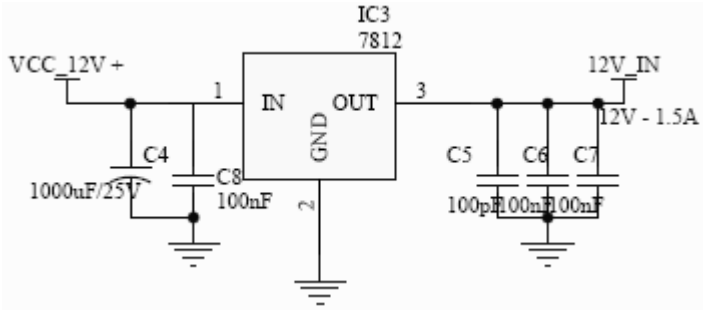

Figure 4.6. Switch regulator circuit for 12V, 1.5A power source.

The L7800 series of three-terminal positive regulators can provide local on-card regulation, eliminating the distribution problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltage and currents.

4.1.4 Relay Part

The PWM source is coming from the microcontroller and is applied over a Relay device. It is an On/Off control by a transistor behaving as a switch. By switching the relay On and Off, it connects or disconnects the 220V AC line. By this way, the heat resistance heats up or cools down.

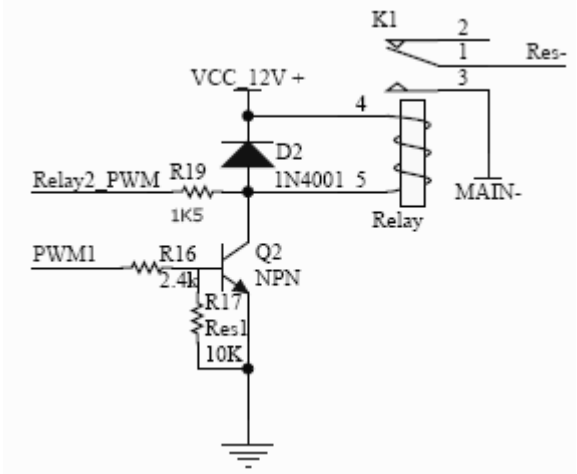

Figure 4.7. Relay circuit of the system to control 220V AC.

Table 4.3. Relay specifications of the system.

Control Voltage	Output Voltage	Output Current
12 V DC	220 V AC	5 A

4.1.5 ST-232 Part

ST-232 device is used as the level shifter for RS-232 communication between the PC and the board. It is particularly suitable for applications where ±12V is not available. The ST-232 uses a single 5V power supply and only four external capacitors (0.1μF).

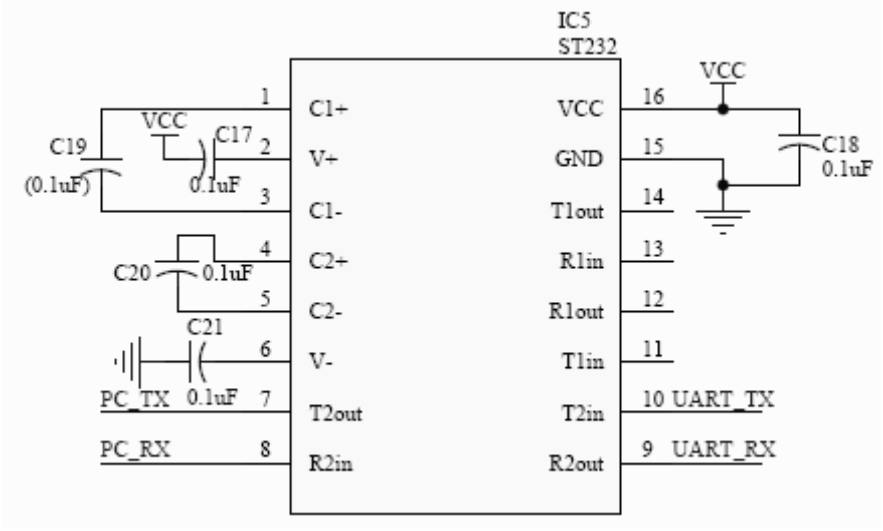

Figure 4.8. ST-232 circuit for RS-232 driver.

4.1.6 AD594 Part

The AD594/AD595 is a complete instrumentation amplifier and thermocouple cold junction compensator on a monolithic chip. It combines an ice point reference with a precalibrated amplifier to produce a high level (10 mV/°C) output directly from a thermocouple signal. Pin-strapping options allow it to be used as a linear amplifier-compensator or as a switched output setpoint controller using either fixed or remote setpoint control. It can be used to amplify its compensation voltage directly, thereby converting it to a stand-alone Celsius transducer with a low impedance voltage output.

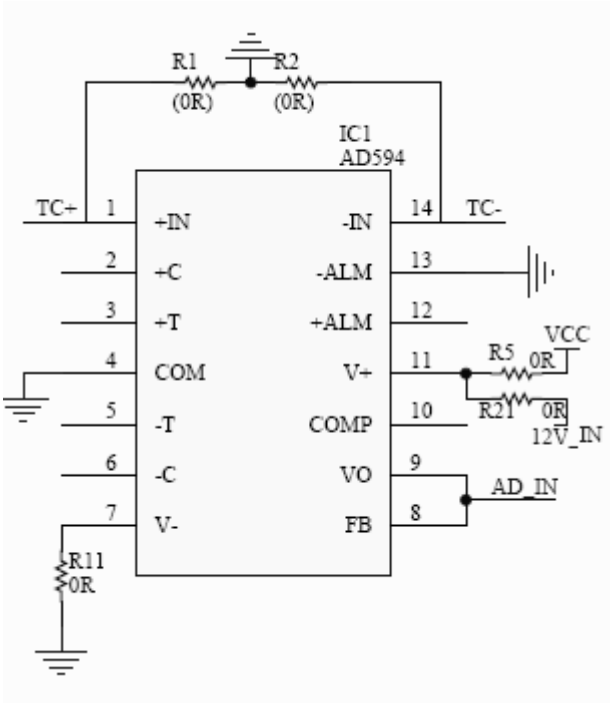

Figure 4.9. AD594 circuit as the thermocouple amplifier.

4.1.7 LCD Module

In order to follow the system on the board, there is an aditional LCD screen connected to the microcontroller. It is 2x8 type and controlled over four data pins. The other control pins are for contrast and data control pins.

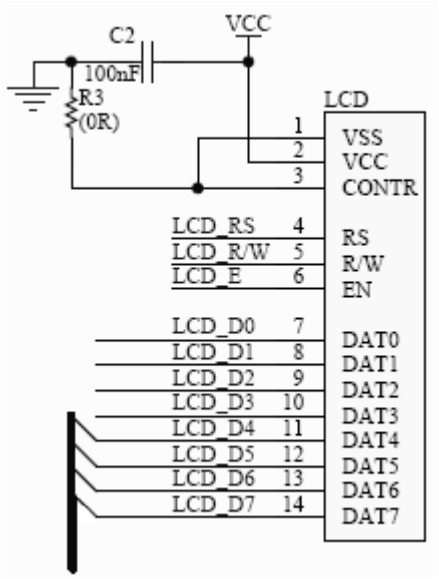

Figure 4.10. LCD circuit to follow the system on the board.

4.1.8 The Microcontroller

The system is fully controlled and driven by PIC18F4520 as the microcontroller of the system. Some features are as follows:

- Single-Supply 5V In-Circuit Serial
- Supports RS-485, RS-232 and LIN 1.2
- 10-Bit, up to 13-Channel Analog-to-Digital (A/D)
- Four Crystal modes, up to 40 MHz
- Supports interrupt on High/Low-Voltage Detection
- Erase/Write Cycle Data EEPROM Memory Typical

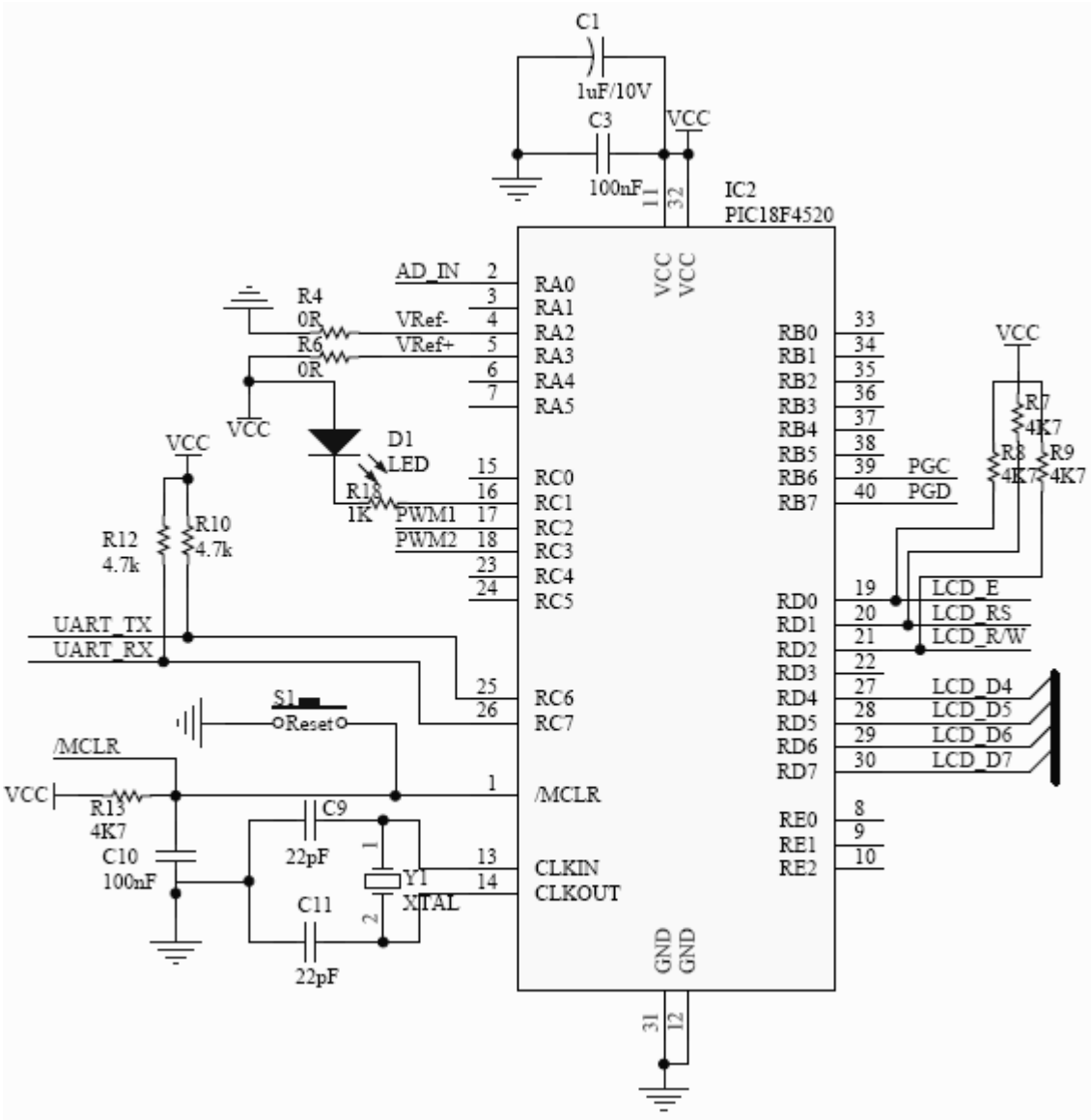

Figure 4.11. PIC18F4520 circuit as the microcontroller of the system.

4.1.9 Thermocouple

A thermocouple is a kind of sensor to measure the temperature. It is pairs of two dissimilar metals joined together at one end. Heating the junction results a voltage corresponding to a temperature degree.

A thermocouple maybe available with different combinations of metals or calibrations. There are four common calibrations which are J, K, T and E types. There are high temperature calibrations R, S, C and GB. Each calibration has a different temperature

range and a maximum temperature value. Actually the maximum temperature depends on the diameter of the wire used in the thermocouple.

Figure 4.12. A basic thermocouple structure

Some criteria to select a thermocouple:

- Temperature range
- Chemical resistance
- Vibration resistance

There are three junction types: grounded, ungrounded, and exposed. A grounded junction offers a good heat transfer from the outside to the thermocouple and a faster response type. On the other hand, the ungrounded type offers electrical isolation.

Table 4.4. Specifications of a J-type thermocouple

Type of Wire	**Temperature Range**	**Standard Grade**	**Premium Grade**
J Type Iron-Constantan	0°C to750°C	Greater of 2.2°C or 0.75 %	Greater of 1.1°C or 0.4 %

4.2. Software Design

4.2.1. General Features

The Control system is fully managed by a unique software module. This software module has been developed for this project and is specific to the control system.

The module has two parts: One part is the Embedded Software which succeeds all the functionalities of the embedded board. The second part is the PC Software which presents a special User Interface to the user and makes all the system functionalitites easily controlled by the user. The user should only select a function on the User Interface and then click the related button, then User Interface will manage all the tasks totgetter with the Embedded part to achive the desired functionality.

The Embedded Software was written in Embedded C programming language while the PC Software was written in Visual Basic. The communicaton between two modules is realized over a serial communication combining with a USB interface.

4.2.2. Embedded Software Module

The Embedded Software module was developed to realize all the functionalities of the embedded board. The software is buried in the microcontroller and can be re-programmed easily on the request. There are many tasks on the this part to realize all the functionalities and all are explained in the following sections.

4.2.2.1. Timers. There are two Timers used for embedded part of the project, TIMER1 and TIMER2. Both timers are 8-bits.

TIMER1 is used to arrange the duty cycle ratio by the counter. The counter is firstly normalized to 100 units corresponding to 100 per cent duty cycle. The duty_cycle parameter is actually set to the exact duty cyclwe value coming from PC. Then, by comparing two values, duty cycle is realised over the control of PWM output port.

$$\text{Clock Interrupt} = 256x0.256 = 65.536\,\text{ms}$$

Maximum duration can be:

$$256x65.5 = 16.768\ \text{s}$$

```
#int_TIMER1
void TIMER1_isr(void) {
        static unsigned int8 counter = 0;
        if(counter < duty_cycle) {
        output_high(PWM1);
        output_low(pin_C1);
        }
        else {
        output_low(PWM1);
        output_high(pin_C1);
        }
        counter++;
        if(counter == pulse_cycle) {
        counter = 0;
        duty_cycle = duty_temp;
        if(duty_cycle <= 3)
        duty_cycle = 0;
        }
}
```

Figure 4.13. Timer1 algorithm

TIMER2 is used to arrange the delay time. Again here, delay_cnt parameter is set by PC by which the delay time is adjusted.

TIMER2 is set to have a clock interrupt of 20 ms. By the way, the maximum duration can be:

$$256x20 = 5.12 \text{ s}$$

```
#int_TIMER2
void  TIMER2_isr(void)
{
  static unsigned int8 cnt = 0;
  cnt++;
  if(cnt < delay_cnt)
    return;
  cnt = 0;
  FRead = 1;
}
```

Figure 4.14. Timer2 algorithm

4.2.2.2.RS-232 Communication. The communication between PC and the Control board is initially realized over RS-232 protocole. Additionally, we are using an RS-232-to-USB Converter to make the project compatible with USB interface. So, the communication is the combination of both RS-232 and USB protocoles.

For the communication speed, we are using 9600 Baudrate. We also use Pin C6 as the transmitter while Pin C7 as the receiver.

Some fundamental functions used during the serial communication are as shown in Table 4.5

Table 4.5. Communication functions and the definitions

Function	**Definiton**
putc (RS232_TResult)	The function sends an 8-bit character over the RS232 XMIT pin.
getc()	This function waits for a character to come in over the RS232 RCV pin and returns the character.
gethex()	This function waits for a Hex to come in over the RS232 RCV pin.

During the operation, there are many transmitted and received signals between the PC and the control board. The main information sent is ADC value coming from the

control board and the Duty Cycle value coming from the PC. This information is regularly shared between the interfaces.

However, there is a problem with RS-232 communication here. Since it is an 8-bit character communication, we cannot sen any value which is greater than 8-bits. Therefore, we have to divide the value into two 8-bits parts and send them separately at the transmitter side. Here, two 8-bits make actually a 16-bits value which is sufficient for all the functionalities of the project. Additionally, we have to combine them to get the real value at the receiver side. Following describes how to realize those two operations.

```
unsigned int16 Result;
unsigned int8 High;
unsigned int8 Low;
High = Result >> 8;
Low = Result & H00FF;
putc(High);
putc(Low);
```

Figure 4.15. Conversion of 16-bits value into two 8-bits values

```
unsigned int16 Result;
unsigned int8 High;
unsigned int8 Low;
High = getc();
Low = getc();
Result = High;
Result = Result << 8;
```

Figure 4.16. Conversion of two 8-bits values into a 16-bits value

The whole system is managed by the starter signal coming from the PC. These signals are received in the RDA part:

#int_RDA

In this section, the incoming signal is switched to the desired mode. It may mean a temperature value to wait for or a signal to start a desired mode. All the incoming signals and the corresponding meanings are shown in Table 4.5.

Table 4.6. List of the signals incoming from the PC and the corresponding meanings.

T	R	S	D	G	M	W	P	F
Time	Reset	Start	Duty Cycle	G(s)	Full Power	PWM	PID Control	Full Control

4.2.2.3. Analog-to-Digital Conversion (ADC). The ADC port has 10-bits resolution for PIC18F4520. Another criteria for the precision of ADC operation is the reference voltage of the microcontroller. In the system, the this voltage is chosen to be 4V.

$$V_{ref} = 4V$$

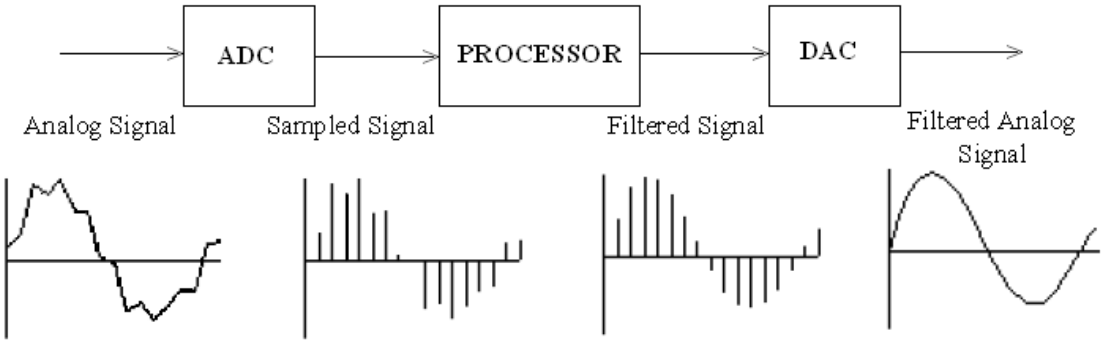

Figure 4.17. Analog to Digital Conversion process to get the desired analog signal at the output

This value can measure up to 400 °C of the temperature. When we combine these criteria to get some information on the system precision, we can easily calculate it by:

$$10 \text{ bits} = 1024 \text{ steps}$$

$$\text{Voltage Precision} = \frac{4000mV}{1024} = 3.910 \text{ mV}$$

Here, 4000mV is the reference voltage in milliVolts and 1024 corresponds the 10-bits value. Since we know that AD594 gives 10 mV/°C ratio, then the temperature precision can be calculated as:

$$\text{Temperature Precision} = \frac{3.91}{10} = 0.391\ ^{\circ}\text{C}$$

This calculation is realized in the function:

```
void ReadandSend_ADC()
```

The function to get the latest ADC value is:

```
ADC_read = read_adc()
```

The function returns a 10-bit value. The array LUT1 keeps the voltage values in order and the array LUT2 keeps the corresponding temperature values. All the information is taken from AD594 datasheet. When a voltage value is to be calculated for the corresponding temperature value, then it is normalized according to those information as in CalcTemp() function and it gives the real temperature value.

```
unsigned int16 ADC_read;
float T_result;
float V_result;
void ReadandSend_ADC() {
        static int rindex = 0;
        int constant, counter;
        float tmp;
        ADC_read = read_adc();
        V_result = ( 4.0/1024.0 ) * ( float ) ADC_read;
        V_result *= 1000;
        T_result = CalcTemp ( V_result );
}
float CalcTemp (float Vx) {
      unsigned int8 i;
      float Tx;
      for (i = 0; i < 11; i++) {
         if (Vx > LUT1[i] && Vx <= LUT1[i+1])
            break;
      }
      Tx  =LUT2[i+1]  -  (((LUT1[i+1]-V_result)*40)  /  (LUT1[i+1]  -
LUT1[i]));
      return(Tx);
}
```

Figure 4.18. Analog-to-Digital and Temperature Calculation Algorithms

4.2.2.4. G(s) Function.

void Calc_Gs()

The function realizes the G(s) functionality of the system. It applies the relay method to get the PID parameters. The operation starts with a PC signal. Then the input from PC is sent which is the temperature value to apply the relay method.

It takes three cycles duration to complete the operation. Since limit cycle occurs with the second cycle, the function starts to calculate the Amplitude and the Period of the cycles after this time. It also measures the variable K_i parameter for the current set temperature.

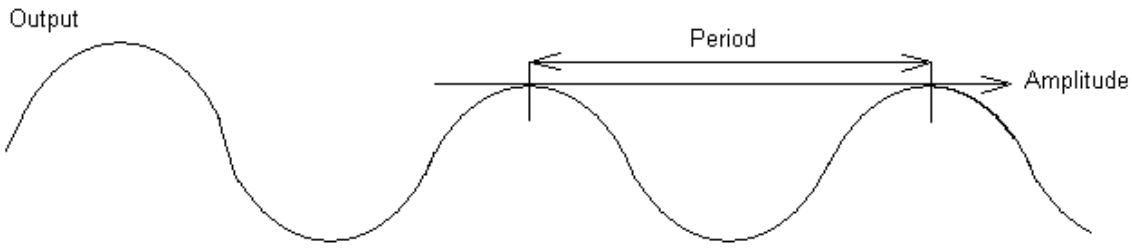

Figure 4.19. Calculation of parameters for G(s) mode

While G(s) function is realizing its operation, it applies a direction finding method to produce the cycles as described in Direction Function section.

4.2.2.5. Direction Function.

void Direction(int value)

Since G(s) functionality needs to get cycles around the set temperature value, the function needs to know where the direction of the output signals go through so that it should change the direction. Also Full Power Function needs to know the direction, as well.

The Direction function finds the direction of the output signals for the purpose above. While realizing the functionality, it takes some series of the signals into consideration and calculates the direction regarding those signals.

The input value to the function tells it which way to check for the direction. While the temperature is increasing, the function should find the point at which the direction gets

a negative value and tells the referencing functions to change the state. It should do the same thing for the positive value in vice versa, either.

4.2.2.6. Full Power Function.

void DeadTemp()

The function realizes the Full Power functionality of the system. The operation again starts with a PC signal. Then the input from PC is sent which is the temperature value to calculate the resulting Full Power Temperature.

The function applies a full power by setting the PWM pin into High state:

output_high(PWM1)

Whenever the output gets the set temperature then, it sets the PWM pin into Low state:

output_low(PWM1);

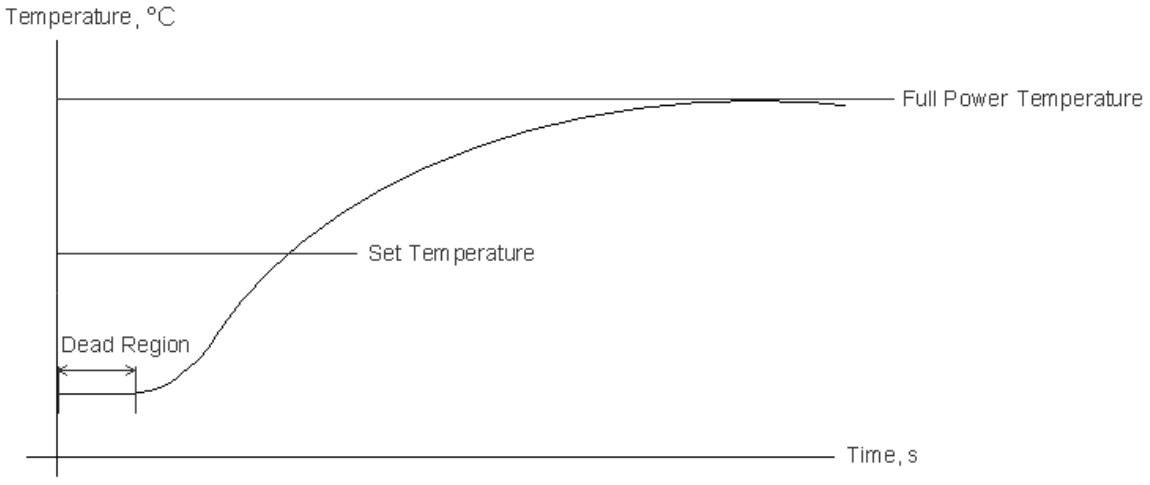

Figure 4.20. Calculation of the Full Power Temperature for Full Power mode based on a set temperature.

Then, the function always keeps the latest maximum temperature value to return. It uses the Direction function to watch the direction of the output signals. Because the aim is to finalize the operation when the direction gets into the negative way. Finally, the function returns the maximum temperature value which is actually the desired Full Power value.

4.2.2.7. PWM Function.

void PWMControl()

The function realizes the PWM functionality of the system. The operation starts with a PC signal. Then the input from PC is sent which is a constant PWM ratio to calculate the resulting PWM Temperature.

The function firstly sets the system duty cycle to the valuecoming from the PC.

duty_temp = duty_cycle = ch

Then it always applies a constant PWM signal to the system. The system settles on a constant temperature value some time later and it returns this value as the PWM temperature.

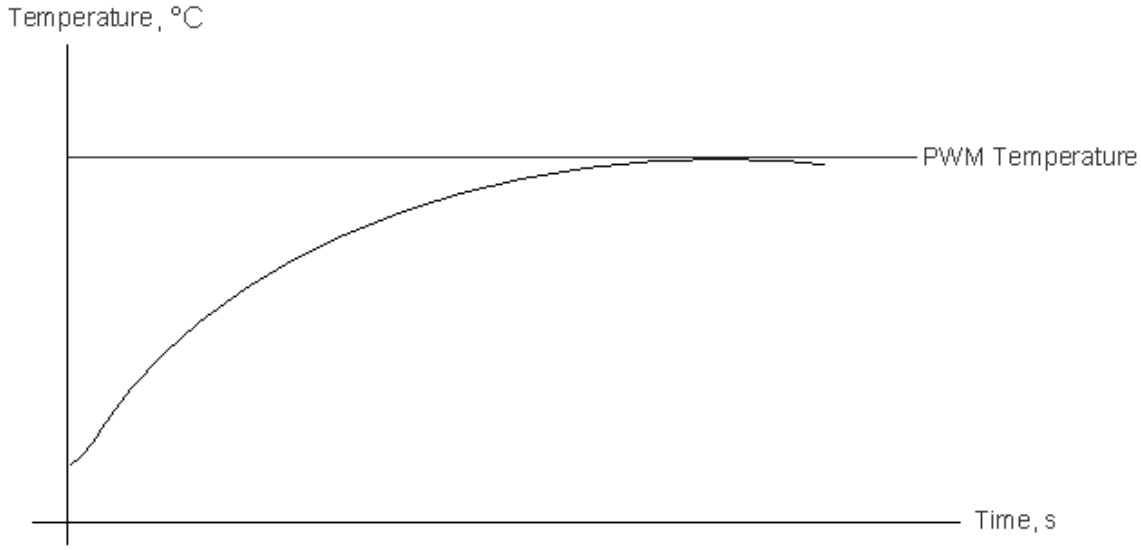

Figure 4.21. Calculation of the PWM Temperature for PWM mode based on a constant PWM input signal.

4.2.2.8. LCD Functions. The system has an LCD interface to show any desired result on the screen. It is a 2x8 character type and written by four data pins. The functions used to write anything on the LCD screen are described below.

printf(lcd_putc, datatype, data)

Function sends the desired data to the LCD. datatype may be %ld, %c or any type and data is the corresponding data to be written on the LCD screen.

void send_LCD()

Function calls series of LCD fuctions to write a group of data to the LCD screen.

4.2.2.9. Main Function.

void main()

This is the main function as included in every software module. It firstly sets up the configurations and then arranges all the functionalities of the system by calling the desired functions separately.

4.2.3. PC Software Module

This module has been developed to present a special User Interface to the user and make all the system functionalitites easily controlled over that. It is very easy to achieve a function on the User Interface. All the functions are shown on the User Interface and the user should only select a function and then click the related button, then the User Interface will realize the rest of the tasks to achive the desired functionality. Figure 4.17 shows the User Interface with all the functions.

The user interface has mainly a graph object at the middle which shows the graph of the temperature values on the screen.

The PC Software module gets ADC values from the Embedded Software module as the inputs and makes some conversions to get the corresponding temperature value. Finally, it shows the result on the screen. It also gets some signals from the Embedded part to check the situation of this part.

buffer = MSComm1.Input

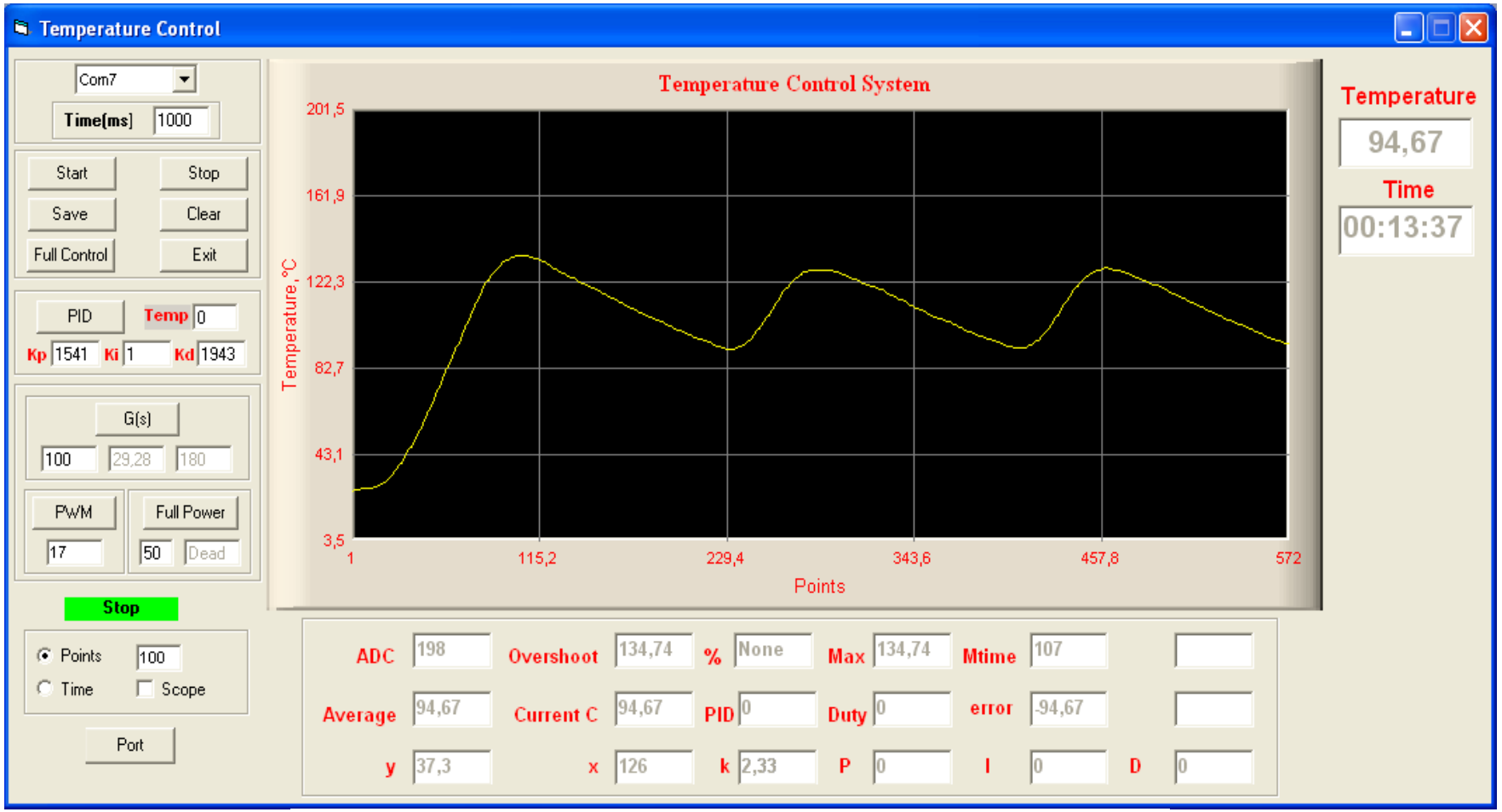

Figure 4.22. The User Interface presented by the PC Software Module.

The PC Software Module also informs the Embedded Software Module which function it should realize by sending a related signal.

MSComm1.Output = data

4.1.1.1. Configuration Functionality.

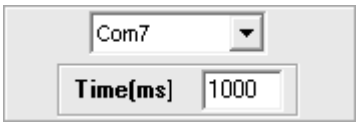

Figure 4.23. Configuration frame

When the system is started, it needs to be configured according to the desired settings. These settings are shown on Figure 4.18. The first configuration is the serial communication port the PC is using to communicate with the Embedded board. The system presents a combobox to select which port to use. Secondly, it shows a textbox to determine the delay time. The delay time is a value in millisecond which is used to arrange the delay between two successive signals. Another setting is the communication settings which are determined in the software module. These settings must suit with the settings in the Embedded module.

MSComm1.Settings = "9600,N,8,1"

4.1.1.2. Controls Functionality.

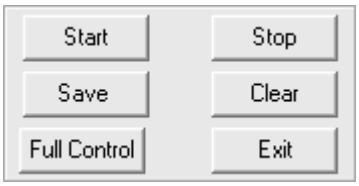

Figure 4.24. Controls frame

- Controls Frame realizes some control inputs on the system.

- Start button starts configuring the settings and initializing the variables.

- Stop button stops the system immediately.
- Save button saves the graph with all the data in csv format.
- Clear button clears the screen.
- Full Control button sets the system into the Full control mode.
- Exit button ends the system and exits the User Interface.

4.1.1.3. PID Functionality.

Figure 4.25. PID Control frame

PID Control Frame realizes the PID functionality of the system. It first informs the Embedded part to get into the PID mode.

MSComm1.Output = "P"

Since PC Module is always informed about the temperature value by the Embedded Module, PID functionality uses this value and applies PID algorithm by using K_p, K_i and K_d values as shown on the frame. The algorithm is shown in Figure 4.26.

```
error = Val(setTemp) - TData
proportion = Kp * error
If proportion > 65535 Then
        proportion = 65535
End If
If proportion < 0 Then
         proportion = 0
End If
differential = Kd * (pre_val - TData)
If differential > 65535 Then
        differential = 65535
End If
If differential < 0 Then
         differential = 0
End If
pre_val = TData
integral = integral + Ki * error
If integral > 65535 Then
        integral = 65535
End If
If integral < 0 Then
         integral = 0
End If
pid_out = proportion + integral + differential
If pid_out > 65535 Then
        pid_out = 65535
End If
If pid_out < 0 Then
        pid_out = 0
End If
```

Figure 4.26. PID Algorithm

In the algorithm, 16-bits values are used and that's why the values are limited between 0 and 65535 against the exceedings. Finally, the algorithm calculates the duty cycle and sends it to the Emdedded Module.

```
duty_cycle = pid_out
duty_cycle = (100 * duty_cycle) / (65535)
duty_cycle = duty_cycle And &HFF
MSComm1.Output = "D"
MSComm1.Output = Chr(duty_cycle)
```

Figure 4.27. Duty Cycle Algorithm

4.1.1.4. G(s) Functionality.

Figure 4.28. G(s) Control frame

G(s) Control Frame realizes the G(s) functionality of the system by using the relay method. It first informs the Embedded part to get into the G(s) mode and sends the desired set temperature value.

```
MSComm1.Output = "G"
MSComm1.Output = Chr((Val(GsTemp) And &HFF00) \ &H100)
MSComm1.Output = Chr(Val(GsTemp) And &HFF)
```

When the G(s) function is finished, then Embedded Module sends the necessary parameters for G(s) functionality which are the Amplitude, Period and variable K_i. Finally, G(s) functionality calculates K_p, K_i and K_d parameters for the PID functionality.

4.1.1.5. PWM Functionality.

Figure 4.29. PWM Control frame

PWM Control Frame realizes the PWM functionality of the system. It first informs the Embedded part to get into the PWM mode and sends the desired set PWM ratio.

```
MSComm1.Output = "W"
MSComm1.Output = Chr(Val(DutyPWM))
```

The operation is finalized by the Embedded module and the desired PWM temperature is found.

4.1.1.6. Full Power Functionality.

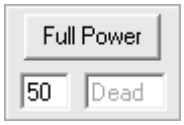

Figure 4.30. Full Power Control frame

Full Power Control Frame realizes the Full Power functionality of the system. It first informs the Embedded part to get into the Full Power mode and sends the desired set temperature value.

```
MSComm1.Output = "M"
MSComm1.Output = Chr((Val(DeadTemp) And &HFF00) \ &H100)
MSComm1.Output = Chr(Val(DeadTemp) And &HFF)
```

The operation is finalized by the signal from Embedded module and the desired Full Power temperature is set on the textbox.

4.1.1.7. Graph Functionality.

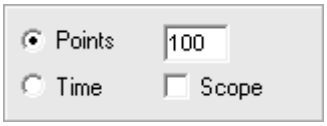

Figure 4.31. Graph Control frame

Graph Control Frame realizes the graph operations on the screen.

Points selection makes the x-axis of the graph in points unit and all the received signals are shown on the screen.

Time selection makes the x-axis of the graph in time(s) unit and the received signals are shown on the screen for every second.

Scope selection shows only a poriton of recently received signals. For example, if the box above is written to be 100, then only the last 100 signals will be shown on the screen.

4.1.1.8. Information Functionality.

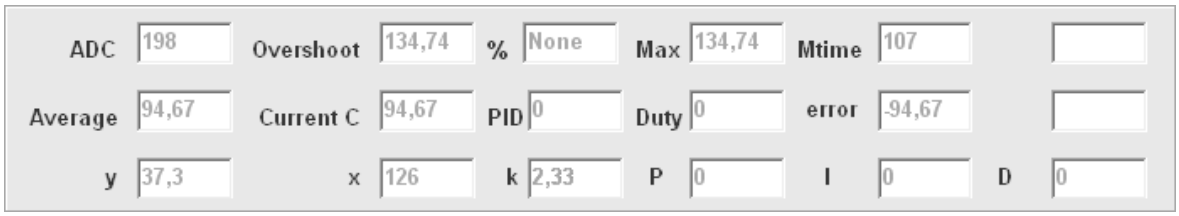

Figure 4.32. Information Control frame

Information Control Frame is to show all the information about the system. The system can be fully watched over this frame. While the static values are shown on the frame like y, x and k values, also the dynamic parameters are shown like temperature, duty cycle, ADC, overshoot, P, I and D values.

1. RESULTS OF EXPERIMENTS

5.1. Full Operation

We consider all the inprovements discussed so far:

- Full Power mode will speed up the system and minimize the overshoot.
- PWM mode will stabilize the system as a supporting input.
- In PID mode:
    i. Changing PID Values will improve the system for different setpoints.
    ii. Set Integral term will reload the PID value and stabilize the system around the setpoint after getting it first.
    iii. Variable K_i will minimize the steady-state error and decrease the settliing time.

After all the improvements, the system output will be smooth as shown in Figure 5.1.

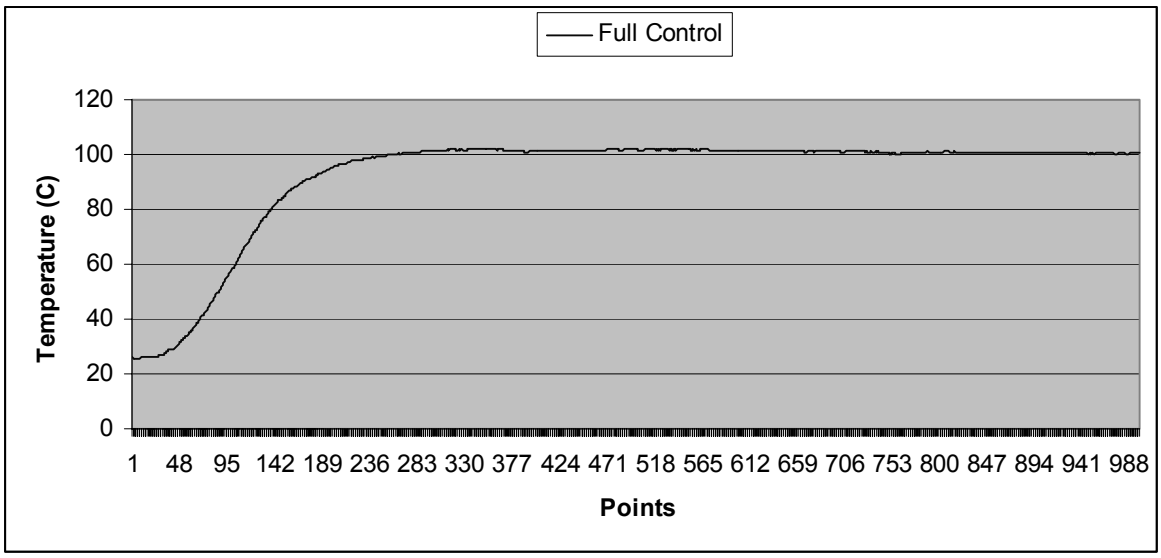

Figure 5.1. Full Control output for 40 W Resistance at 100 °C

The duty cycle output of the system is given in Figure 5.2. While Table 5.1 is showing the parameters for each mode during the Full Control System, Table 5.2 shows Full Control System outputs of Figure 5.1 for 40W Resistance at 100 °C.

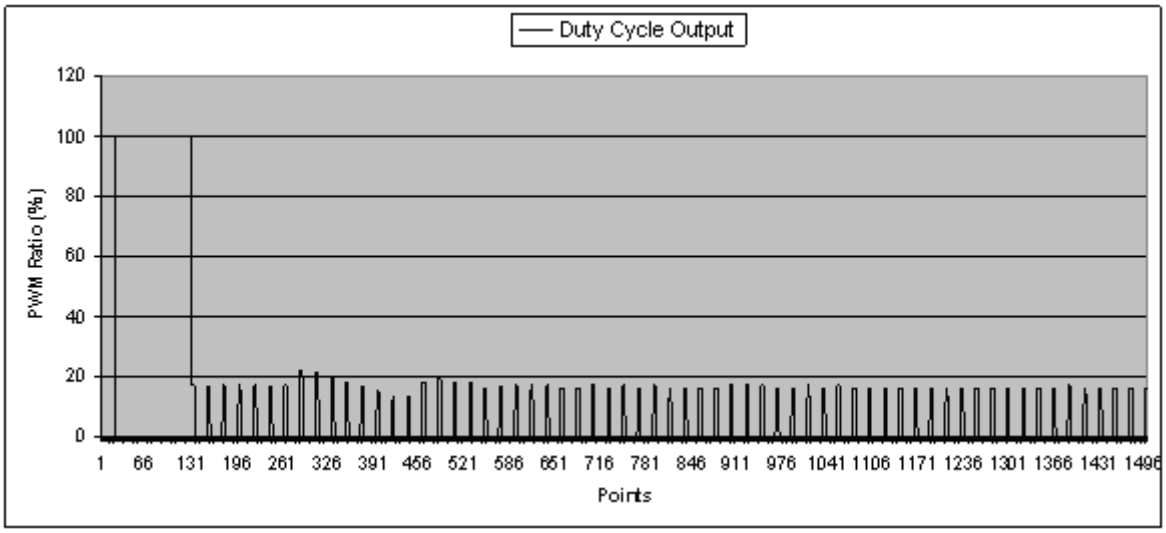

Figure 5.2. Duty Cycle Output for 40 W Resistance at 100 °C

Table 5.1. Parameters for each mode during the Full Control System of Figure 5.1 for 40W Resistance at 100 °C

Full Power(°C)	**PWM(per cent)**	**P**	**I**	**D**
52	15	18.45	0.011	21.6

Table 5.2. Full Control System outputs of Figure 5.1 for 40W Resistance at 100 °C

	Maximum Temperature(°C)	**Overshoot (per cent)**	**Seetling time(s)**	**Number of undershoots**
Full Control	101.62	1.62	208	0

Finally, we get an improved system with:

- Minimized Overshoot
- Minimized Undershoot
- Minimized Setling Time
- Minimized steady-state error

The comparison of the Standard PID, PID with Variable K_i and Full Control experiments are shown in Figure 5.3.

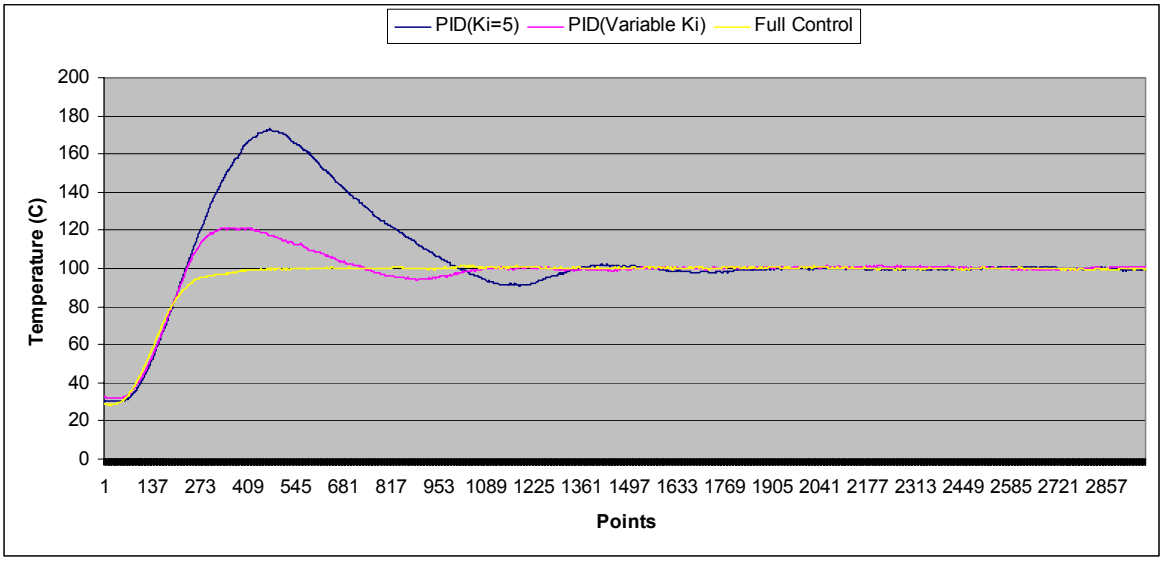

Figure 5.3. Comparison of the control outputs for 40 W Resistance at 100 °C

5.2. Multiple Setpoints

We have been working on single point operations so far. That is, we set a single set value, auto-tuned the system and applied the conditions according to our Full Control rules. These works so far showed us that the proposed Full Control design can improve the system well with all the aspects.

In this chapter, we will apply multiple set points rather than only one and will make all the tests again to see the system performance for a more generalised and more complex structure.

Since we have several set values now, then we have to adapt the system to that by applying some other rules. That is because, for example, we may apply Full Power mode only for the first set value in some conditions. Therefore, we divide the system into two part: first set value and others. Assume that we have some set values which some are at higher temperatuıre values, some are at lower that makes the system up and down

branching. Then we apply the following rules to adapt the system working in a multiple-points mode.

- Autotune the system covering these set values.
- Load Full Power, PWM and PID mode tuned parameters.
- Load the set values and the corresponding desired time durations for each one.
- Initialise the system with Full Power mode aiming the first set value.
- Swicth to PWM mode just after the Full Power mode.
- Then, start the PID mode by setting the integral term as described in Section 3.4.2 and go on this mode to stabilise along the set value.
- When pre-set time duration is up for the first set value, then switch to the second set value operation but by partitioning the first and the second set value interval into smaller ones. Here, we actually divide the interval into more set values with suitable time durations for each one and then continue the operation. This rule is importantly necessary to reduce the overshoot for the next set value.
- At the same time, check the PID parameters to find and set the best ones for the current set point as described in Section 3.4.1. Then, PID parameters will dynamically change for the best performance.
- When the system gets the second (the real) set value, then stay there and stabilize the system on the value.
- If next set value is lower than the current one, then fully shut down the power to leave the system cool down through the new value.
- Then, when the system approaches to the set value by 10 per cent, start the PID mode by setting the integral term as described in Section 3.4.2. This rule is importantly necessary to reduce the undershoot for the next set value.
- Finally, continue with PID mode by taking the time durations into account and switch to new set points if there are

After all the rules described above, consider an experiment of which the output is shown in Figure 5.4. Here, there are three set points, namely 100 °C, 150 °C and 80 °C, respectively.

The set points with the desired time durations are shown in Table 5.3. Here, it seems that there are six set points but actually there are three points with double indication. The second repetition of each point is for the transition region from one point to another.

Table 5.3. System set points for a Multiple Points Full Control experiment

Set Temperature (°C)	100	100	150	150	80	80
Time Duration (S)	100	350	300	300	300	300

The tuned parameters for the experiment are shown in the following tables. While Table 5.4 is showing the PID tuning parameters, Table 5.5 shows the Full Power tuning parameters. Finally, Table 5.6 shows the PWM tuning parameters.

Table 5.4. PID tuning parameters for Table 5.3

	80 °C	**100 °C**	**150 °C**
$\mathbf{K_p}$	17.05	18.45	18.64
$\mathbf{K_{i1}}$	0.01	0.011	0.014
$\mathbf{K_{i2}}$	0.021	0.022	0.020
$\mathbf{K_d}$	23.88	21.6	17.04

Table 5.5. Full Power tuning parameters for Table 5.3.

Final Temperature(°C)	80	100	140	160	200
Full Power Temperature	45	60	110	140	180

Table 5.6. PWM tuning parameters for Table 5.3.

Final Temperature(°C)	60	80	100	150	200
Constant PWM ratio	9	13	18	30	40

After loading the set points and the tuned parameters, initialising the system and applying the rules above, we get the system output as shown in Figure 5.4.

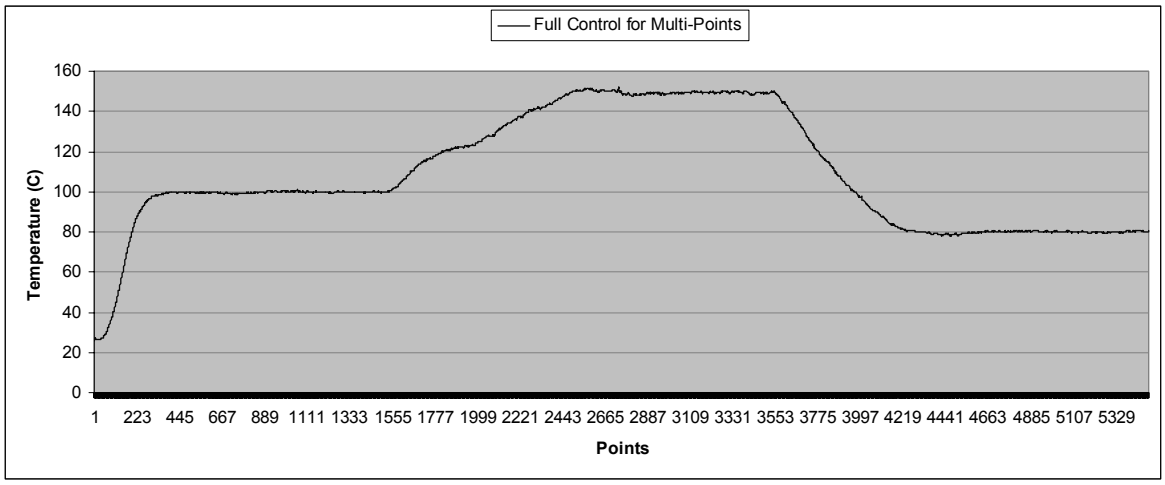

Figure 5.4. Multiple Points Full Control Output for 100 °C, 150 °C and 80 °C set values, respectively.

Figure 5.5 shows the corresponding Duty Cycle Output for Figure 5.4. As it is shown that the output is constant during the regions where the current temperature is on the set temperature, as expected.

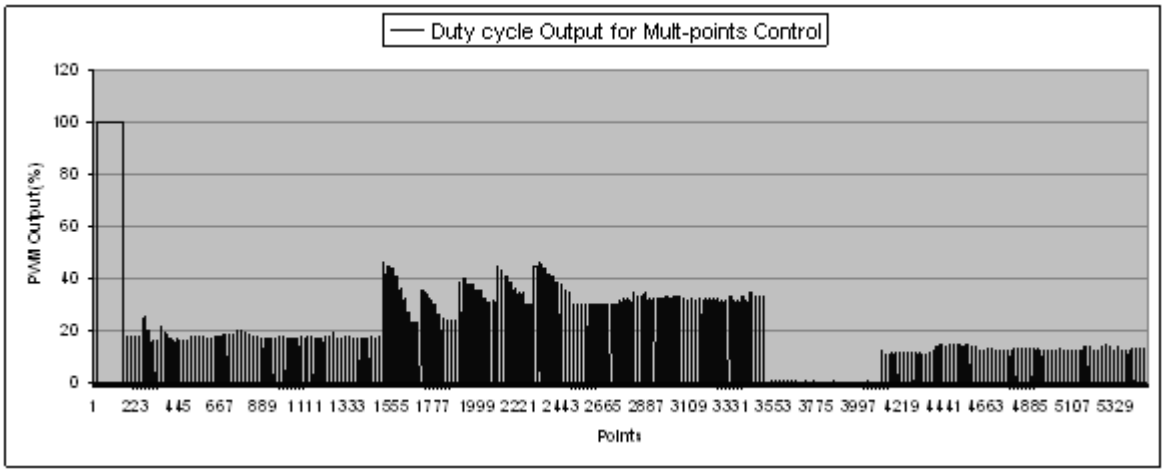

Figure 5.5. Duty Cycle Output for Figure 5.4.

5.3. Calculations

The experiments shown in the figures were realized with the period of 6.5 seconds and the sampling frequency of 3.3 Hz. All the calculations are based on these criteria.

5.3.1. Calculation of PID parameters for 100 °C

Consider Table 3.6 to calculate the PID parameters in. First of all, we get the necessary information and keep them in Table 5.7.

Table 5.7. Relay method parameters for 40 W Resistance at 100 °C

A	D	T_c
29	700	180

Now, calculate the ultimate gain of the system by using the Equation 2.1.

$$K_c = \frac{4\mathrm{d}}{a\pi} \text{ where } a = 29 \text{ and } d = 700 \text{ where } K_c = K_u .$$

$$K_c = \frac{4 \times 700}{29 \times 3.14} = 30.75$$

Calculate the PID parameters by using the equations in Table 2.1.

$$K_p = 0.6K_c = 0.6 \times 30.75 = 18.45$$

$$K_i = \frac{2}{T_c} = \frac{2}{180} = 0.011$$

$$K_d = 0.12T_c = 0.12 \times 180 = 21.6$$

5.3.2. Calculation of PID parameters for 80 °C

Consider the PID parameters for 80 °C shown in Table 3.6 and then continue with the calculations below.

Table 5.8. Relay method parameters for 40 W Resistance at 80 °C

A	D	T_c
31.38	700	199

$$K_c = \frac{4\mathrm{d}}{a\pi} \text{ where } a = 31.38 \text{ and } d = 700.$$

$$K_c = \frac{4 \times 700}{31.38 \times 3.14} = 28.42$$

Calculate PID parameters by using the equations in Table 2.2:

$$K_p = 0.6K_c = 0.6 \times 28.42 = 17.05$$

$$K_i = \frac{2}{T_c} = \frac{2}{199} = 0.010$$

$$K_d = 0.12T_c = 0.12 \times 199 = 23.88$$

5.3.3. Calculation of PID parameters for 150 °C

Consider the PID parameters for 150 °C shown in Table 3.6 and then continue with the calculations below.

Table 5.9. Relay method parameters for 40 W Resistance at 150 °C

A	D	T_c
28.7	700	142

$$K_c = \frac{4\mathrm{d}}{a\pi} \text{ where } a = 28.7 \text{ and } d = 700.$$

$$K_c = \frac{4 \times 700}{28.7 \times 3.14} = 31.07$$

Calculate PID parameters by using the equations in Table 2.2:

$$K_p = 0.6K_c = 0.6 \times 31.07 = 18.64$$

$$K_i = \frac{2}{T_c} = \frac{2}{142} = 0.014$$

$$K_d = 0.12T_c = 0.12 \times 142 = 17.04$$

5.3.4. Calculation of Clock Interrupt for TIMER1

We calculate one complete clock for TIMER1 below.

$$\text{Input Clock} = 4 \text{ MHz}$$

$$\text{Output Clock} = \frac{4\,\text{MHz}}{4} = 1\text{MHz} = \frac{1}{1000000}\,\text{sec}$$

$$\text{One single Interrupt} = 256x\frac{1}{1000000} = 0.256\,\text{ms}$$

Since TIMER results one complete interrupt for each 256 single interrupts then;

$$\text{Clock Interrupt} = 256x0.256 = 65.536\,\text{ms}$$

Maximum duration can be:

$$256x65.5 = 16.768 \text{ s}$$

2. CONCLUSIONS

Asymmetrical processes are complex systems with different behaviors for different modes. Standard methods with a single set of parameters are not suitable to overcome the asymmetrical problems and achieve the desired performance. Therefore, engineers have been working on developing new methods to improve the control of such processes.

Temperature control systems are the most commonly used asymmetrical processes in the industry. The degree of asymmetry for those systems is more severe. Therefore, controlling of such processes is difficult with the standard methods.

In order to handle with the problems for such systems, we worked on an experimental system and developed a new model that applies a 4-mode controller design so that each one improves different effect of the system. In this model, we applied a Full Power mode to handle with the dead band characteristic of the system and it minimized the overshoot while reducing the settling time. Additionally, we supported the system by PWM mode that helped the system to settle on the setpoint and cancel the offset. G(s) mode supplied the PID parameters according to the desired temperature and it gave some important information about the asymmetrical structure of the system. This information is actually the speed of the system dynamics in heating and cooling modes so that we used it on the calculation of K_i to fix the problems of the asymmetrical system behavior. As the main operation of the model, we applied a PID mode that offers a few innovative methods. Setting integral value played a bridge role between PWM and PID modes and effectively drove the system to the setpoint. As a novel approach, variable K_i balanced the system dynamics on two modes by differentially loading the system. The experiments showed that this approach minimized the asymmetrical effect on the system by loading the system by different amounts in two heating and cooling modes. The approach improved the system during the steady-state and minimized the steady-state error.

The model discussed so far resulted an improved system with:

- Minimized Overshoot
- Minimized Undershoot
- Minimized Setling Time
- Minimized steady-state error

During the experiments, we observed that derivative term (D) has little impact on Temperature Control Systems since the temperature changes gradually and so error is small which makes the derivative term affect the system with small value. Therefore for the systems with gradually changing structure may prefer PI controllers for the simplicity.

On the other hand, we have seen some troubles with the study that may be improved. Firstly, since the project applies a 4-mode tuning method, tuning takes some long time. Most of the time is taken during PWM and Full Power tuning modes. There may be some other approaches to shorten this time. Additionally, we mostly worked on the steady-state region during our study and improved the asymmetrical structure of the systems. Together with this region, there may be some other researches which consider the deadband characteristics of the system to get some general numerical methods and combine them with the proposed methods discussed in this study.

APPENDIX A: PROJECT HARDWARE SCHEMATIC

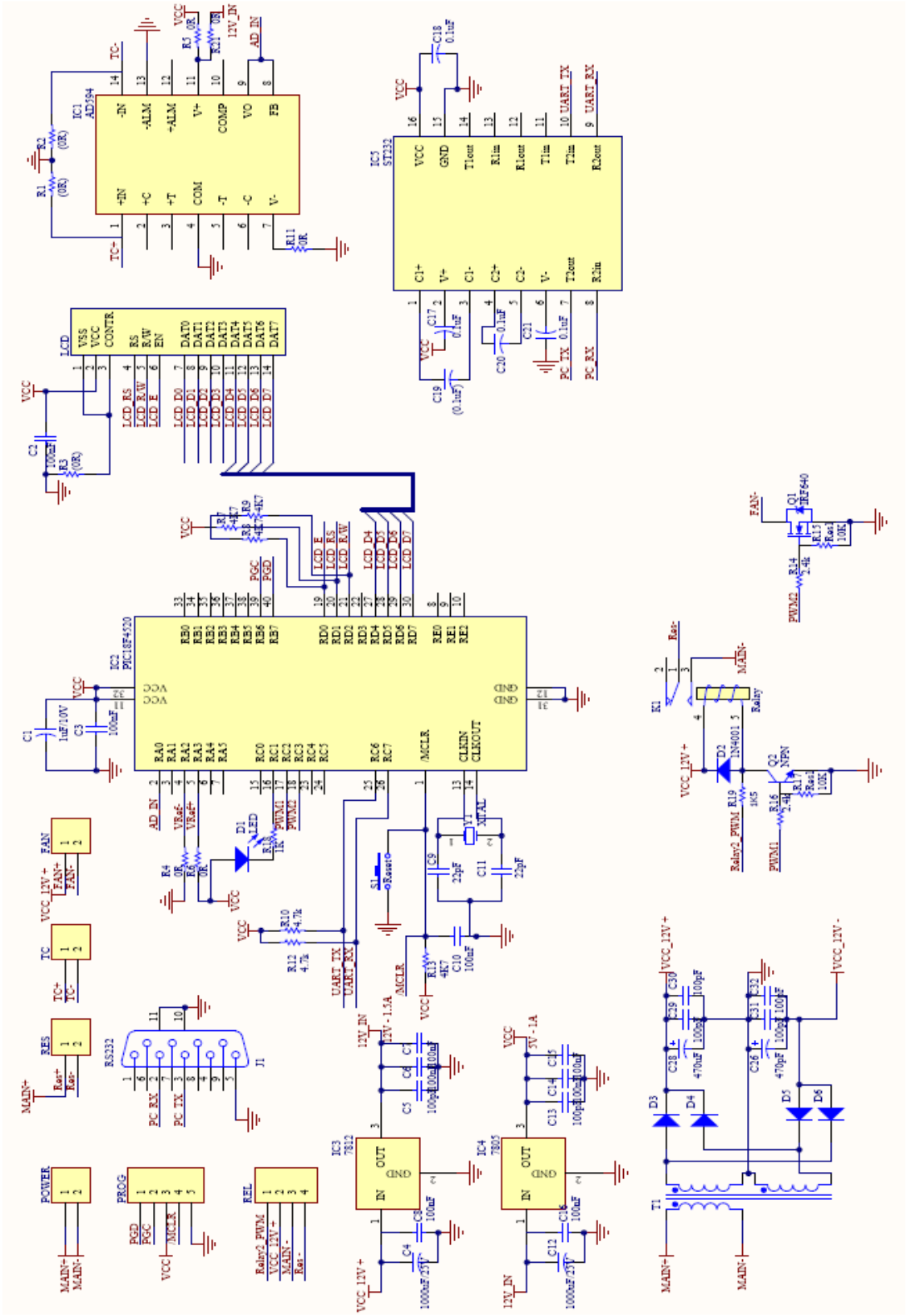

Figure A.1. Project Hardware Schematic

APPENDIX B: PROJECT PCB LAYOUT

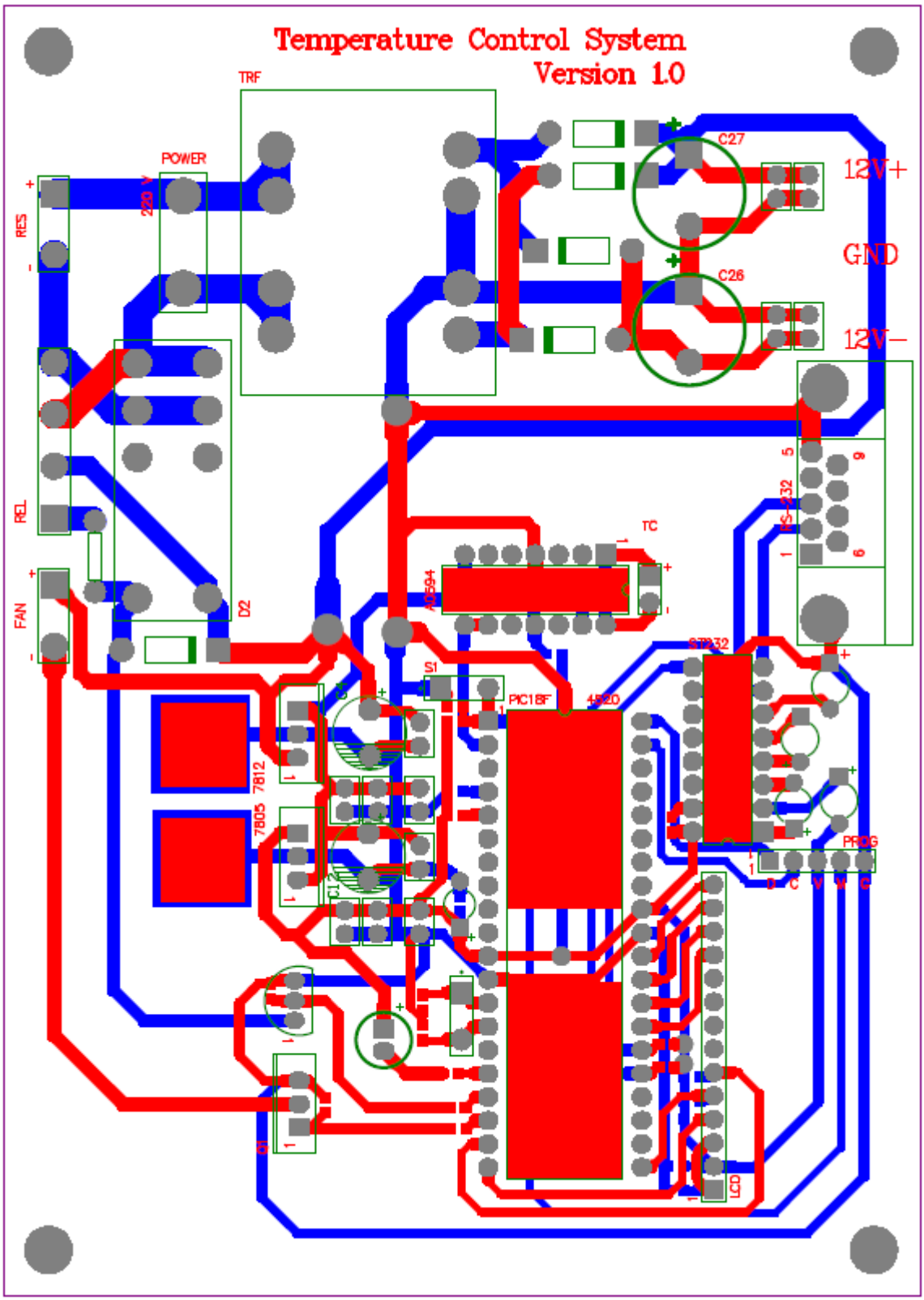

Figure B.1. Project PCB Layout

REFERENCES

1. Tan, Kok Kiong, Wang Qing-Guo, Hang Chang Chieh. Advances in PID Control, Springer-Verlag, London, 1999.

2. Karl Johan Aström, Control System Design, Department of Automatic Control, Lund Institute of Technology, 2002.

3. Aström, K. J., and T. Hägglund, PID Controllers: Theory, Design, and Tuning, Instrument Society of America, Research Triangle Park, NC, 1995.

4. E. Doğru Bolat, Y.Bolat, K.Erkan, Temperature Control Using Improved Autotuning PID Control Methods, ELECO'99 International Conference on Electrical and Electronics Engineering, E01.57/C-12, 1999.

5. K.K. Tan, Q.-G. Wan, Wang, T.H. Lee, C.H. Gan, Automatic tuning of gain-scheduled control for asymmetrical processes, Department of Electrical Engineering, Elsevier Science Ltd, June 1998, 0967-0661(98)00091-4.

6. Shih-Haur Shen, Jiun-Sheng Wu, and Cheng-Ching Yu, Use of Biased-Relay Feedback for System Identification, AIChE Journal, April 1996, Vol. 42, No. 4.

7. Hagglung, T. and A. Tengvall, An automatic tuning procedure for unsymmetrical processes. Proceedings of European Control Conference, Roma, September, 1995.

8. Qing-Guo Wang, Chang-Chieh Hang, Wei Zou, Automatic tuning of nonlinear PID controllers for unsymmetrical processes, Elsevier Science Ltd. Computers chem. Engng Vol.22, No.4/5, pp. 687-694, 1998.