CSE 3215 Embedded Systems Laboratory
Lab 5 Digital Control System

Introduction

The purpose of this lab is to introduce you to digital control systems. The most basic function of a control system is to take a measurement of the current state, and through a combination of software and hardware drive a system to the desired state. Advanced control theory is quite complex and relies on advanced mathematical techniques. In this lab we will analyze a simple first order thermal system and learn how to implement a feedback controller.

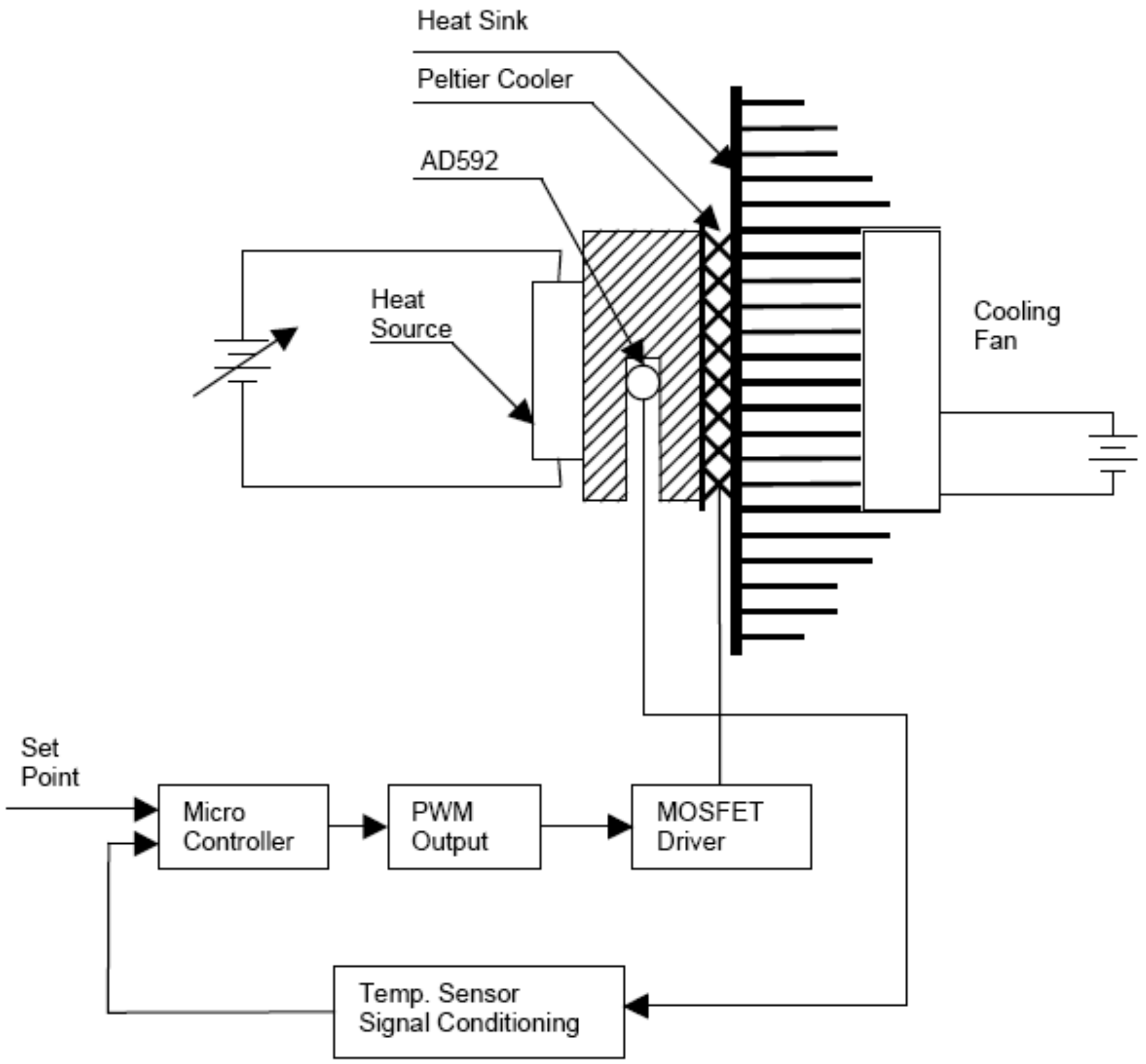

Figure 5.1 Temperature control system

Your task will be to design a ***PI*** (Proportional, Integral) control based temperature controller using the Dragon12 board that will maintain the temperature of an object at 24+/-0.1°C. Figure 5.1 shows a block diagram of control system for our temperature control process.

Control Systems

A control system is a system, electrical, mechanical or otherwise designed to maintain a process variable (for instance the speed of a motor, temperature in a boiler, …). A feedback control system has a means of varying the process variable, and some method of determining the current value of the process variable (a ‘feedback’ signal). The feedback signal is compared to the desired value of the process variable (‘the set point’) and used to adjust the process variable to reduce the error between desired and actual value of the variable.

Our temperature controller attempts to maintain a set point temperature in a block of aluminum despite the fact that it is being heated by a resistive heater. This is analogous to the problem of maintaining acceptable CPU temperatures based upon measurements of package temperature.

The controller uses an electrical heater/cooler known as a Peltier Cooler to vary the temperature of the block of aluminum. A semiconductor temperature sensor, the AD592, is thermally coupled to the aluminum block to determine the current value of the process variable, the temperature. A typical block diagram of a control system in its simplest form is shown in Figure 5.2.

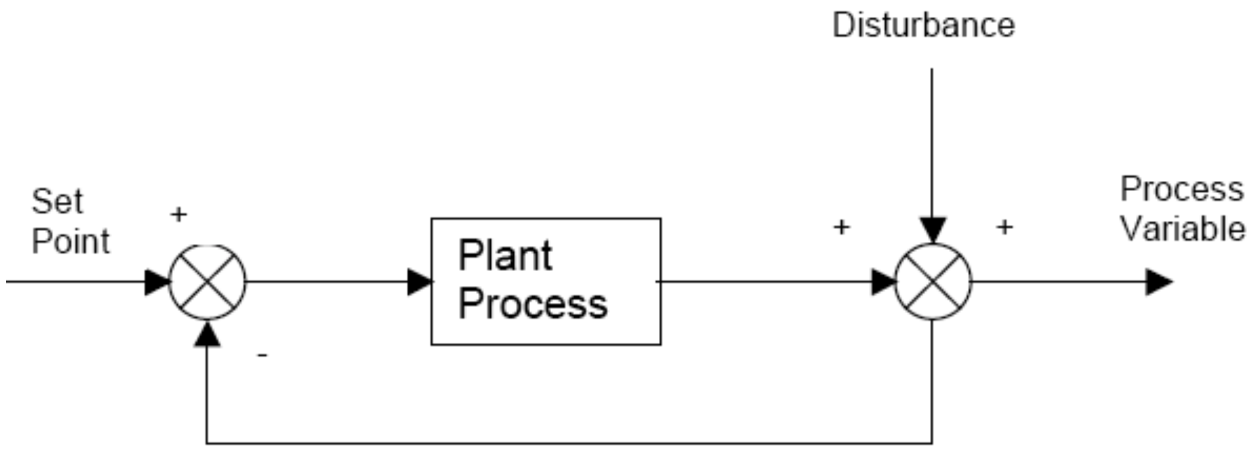

Figure 5.2 Typical plant process control system

A control system can be classified by how it quickly it can respond to a change in set point (or to an outside disturbance). The system is “critically damped” when the system response to a step change in the set point is quick with no oscillations. The system is said to be “over damped” when the response to the change is slow compared to the natural response of the plant process. The system is “under damped” when, although the response to the change is quick, the process variable overshoots and oscillates around the set point before finally settling down. The purpose of the *PI* controller is to tune the system to a point where the response is quick and with minimal overshoot and oscillations.

PI Controller

Mathematically a PI controller can be represented as:

$$u(t) = K_P e(t) + K_I \int_0^t e(t)\, dt$$

where:
$u(t)$ = Controller output signal
$e(t)$ = Controller error
K_P = Proportional gain
K_I = Integral gain

The pseudo code for the above PI equation can be as follows:

```
Begin
    Define set point: $\Theta_{set}$
    Do Forever
        Measure system output: $y(t)$
        Calculate the error: $e(t) = \Theta_{set} - y(t)$
        Calculate the Integral term: $p(t) = K_I \times e(t) + p(t-1)$
        Calculate the PID output: $u(t) = K_P \times e(t) + p(t)$
        Update Control variable
        Save variable: $p(t-1) = p(t)$

    End Do
End
```

Here ***t*** represents time. The time difference between ***t-1*** and ***t*** is the sampling interval.

Design Specification

The program must function as follows:

1. The temperature controller has an adjustable setting from 20.0°C to 28.0°C with a default setting of 24.0°C. VR2 on the Dragon12 is used to vary the set point.

2. The temperature is monitored continuously using channel 6 of the onboard ATD converter and displayed on the 4-Digit 7-Segment display. Pressing push-button SW2 displays the set point temperature.

3. The temperature is controlled by varying the current through the Peltier Cooler using the pulse width modulated output (channel 7) of the HCS12. Peltier Coolers are semiconductor devices that transfer heat from one surface to the other when supplied with a current.

Pre Lab

Develop a program according to the design specifications. The software can be readily decomposed into functional components:
1) Temperature measurement.
2) Peltier Cooler current control.
3) Temperature and set point display.
4) Set point adjustment.
5) Temperature control.

1. The output of the temperature sensor-conditioned amplifier (hardware provided in the lab) must be connected to the ATD channel 6 of Dragon12. The ATD must be configured as 10-bit unsigned (input signal range between 0 and 5.12 Volts). The digitized voltage code is (V_0). Develop your temperature control program to read the voltage level on ATD channel 6 **under interrupt**, convert it to ºC and display the temperature value precise to 0.1ºC on the 4-Digit 7-Segment display, again **under interrupt**. You should average several readings to get smoother measurements. Note the temperature in degrees Celsius is calculated as follows: $T = V_0 /10.$

2. Varying the current through the Peltier Cooler controls the temperature. Setup your control program to output a free 118 Hz signal on PWM channel 7. Adjusting the duty cycle of this signal (e.g. between 0% and 50%) controls the current.

3. The actual temperature and the set point are both displayed on the four-digit 7-segment display. The default display is the actual temperature. The set point is displayed while actuating the momentary switch SW2.

4. The set point is adjustable from 20.0°C to 28.0°C. Design your program to adjust the set point using the potentiometer VR2 on the DRAGON12 display. Once again you

should average several readings using the ATD peripheral and get smoother measurements. (Hint: one possibility is to have a timer that gives an interrupt every 1 ms to time the display update. After ***n*** display updates (where ***n*** depends on your sampling period), you could initiate an ATD conversion that will generate an interrupt upon completion. You could maintain a counter in the timer ISR, to keep track of the number of display updates and thus know when to do the conversions. The mainline can concentrate on controlling the temperature without having to worry about the details).

5. The main function of your temperature control module is to sample the temperature at a regular interval, and use the measured temperature values to update the PWM duty cycle. Implement a method of generating the sampling interval, and perform the PWM duty cycle calculations based on the algorithm for the Parallel ***PI*** controller. (Hint: A sampling period of 3 seconds seems to work well.)

6. Read the lab procedure. You will be making several measurements to characterize the devices used. You should develop code and strategies to facilitate this process. You should also develop your software so that you can test all the functional components separately.

In Lab

Note: Ensure that you are wearing your antistatic wrist strap prior to handling the DRAGON12 board. Failure to do so could result in severe damage to the DRAGON12.

Make appropriate notes and answer all questions clearly in your lab book.

The Connection diagram is shown in Fig. 5.1.A. For the heater and thermolelectrical (Peltier) cooler the two regulated sections of PS-3330 power supply are used. Observe the polarity connections for the battery power supply and the Peltier cooler. Adjust the voltage to the heater by measuring it directly on the heating resistor using a digital voltmeter. Adjust the current through the Peltier cooler using the panel amperemeter of the PS-3330 at 50% duty cycle of the PWM control signal.

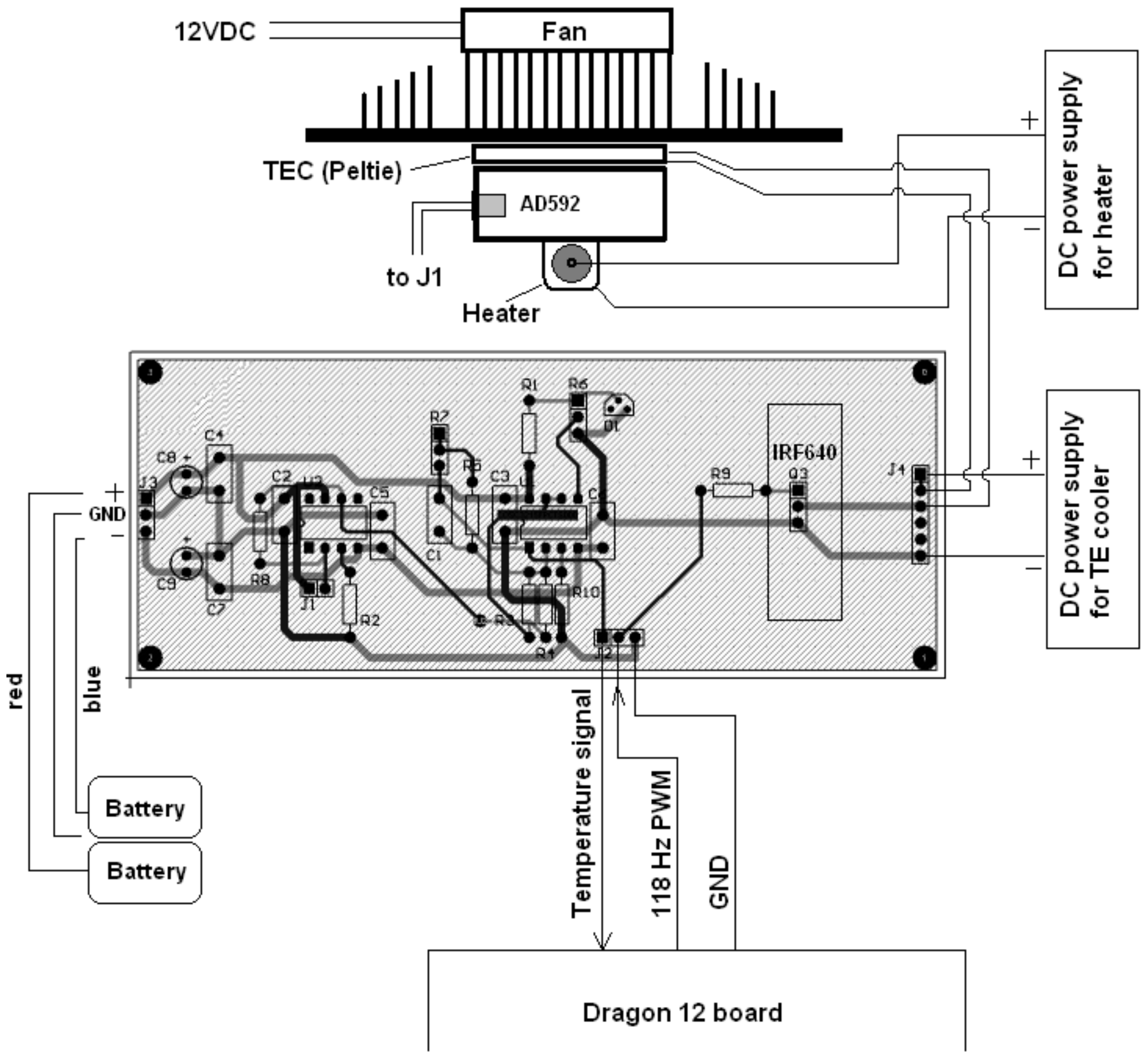

Fig. 5.3. Connection diagram

Part 1 Measuring the Open Loop Response

1. Have your prelab assignments approved by the T.A. prior to commencing the lab.

2. Once you have approval from the T. A. to continue, download and debug your assembled or compiled programs. Once you feel comfortable that your programs are working move on to the experiments below. Before making any electrical connections turn ON the power supply PS-3330 and verify that the two sections are in Independent mode. Adjust initially the voltage for the heater at O V and the voltage for Peltier cooler

at about 3 V. Put the current limit dial for the Peltier cooler at middle position. Turn OFF the PS-3330 power supply. Make all connections according to Fig. 5.3.

3. Verify your PWM frequency is 118 Hz and adjust the duty cycle to 50%. The pulse amplitude must be no less than 4.3 V.

4. Turn ON the power supply of the fan and the PS-3330. Adjust the current through the Peltier cooler to 1.3 A by gradually increasing the voltage. Ensure the power supply is not going into current limiting. Note this voltage. Wait for a few minutes the temperature measured by the temperature sensor to stabilize and note the temperature.

5. Set the PWM duty cycle to 0%. Verify that the current through the Peltier cooler is zero.

6. Adjust the voltage to the power resistor, the heat source, to give a steady state temperature of the device of 34.00 C. The approximate voltage on the heat source's power supply is listed in Table 5.1 for each device number.

7. Once you have attained a stable temperature vary the PWM duty cycle between 0% and 50% using the (e.g. using the DIP switch onboard DRAGON12). Measure the temperature/duty-cycle relationship of at least eight different duty cycles.

a. What type of relationship was there between duty cycle and temperature?
b. Was there any indication of hysteresis?

8. Using your observations above implement an open loop controller (i.e. mapping from VR2 set point to PWM output) that attempts to set the temperature of the aluminum block as close as possible to 24 C. Demonstrate your controller to the TA.

Part 2 Implement the Closed Loop *PI* controller

1. Adjust the set point using the potentiometer VR2 to setting of 24.0°C. Pressing SW2 should toggle the 7 segment display between the set point and the actual temperatures.

2. Tune the Proportional and Integral gain constants of your program, using the details provided below to bring the actual temperature of the aluminum block to 24.0°C. Let the temperature stabilize.

3. Make an adjustments to the set point and measure the temporal response of your controller to this change.

Demonstrate your system to the TA.

Evaluation

This is a two-week lab; there will be no grade for week 1 unless you have completed the entire lab. You must demonstrate a working version of your code to the lab demonstrator (note this means you must show convincing evidence that it works) and have acceptable written documentation about your programs performance. You should be able to answer questions about the program, the use of the test equipment and your test and debugging techniques. The performance of your controllers and the results of the lab experiments should be neatly and fully documented and you should be prepared to answer questions about these results.

Reference Reading

PWM user's guide

Enhanced Capture Timer user's guide (S12ECT16B8V1.pdf)

ATD Converter user's guide (S12ATD10B8CV2.pdf)

Data Sheets for Analog devices AD592AN Temperature Sensor(AD592.pdf)

Dragon12 schematics

Unit Number	**34° C Power Setting (V) (nominal)**
1	**10.5**
2	**10.7**
3	**12.7**
4	**13.8**
5	**9.5**
6	**13.5**

Table 5.1: Resistive Heater Power Settings. Note these are *nominal* values.

Hints

Temperature Measurement

The quickest way to measure the temperature and the temporal response is to send the temperature readings to the console using the *DeBug12 PutChr* function at a known time interval. This information can then be copied or logged from the HyperTerminal communications window and pasted into *MatLab* (preferred) or *OpenOffice.* Subsequent graphs can be drawn in either program. The temperature sensor and the cooler have a long thermal time constant. Remember to allow some time for settling when doing your measurements.

Tuning the PI controller

Figure 5.2 shows the transient response of the temperature controller for certain values of K_P and K_I. Your ***PI*** controller will attempt to reduce the rise time and overshoot while trimming down the steady state error to within 0.1° C of the set point (24° C).

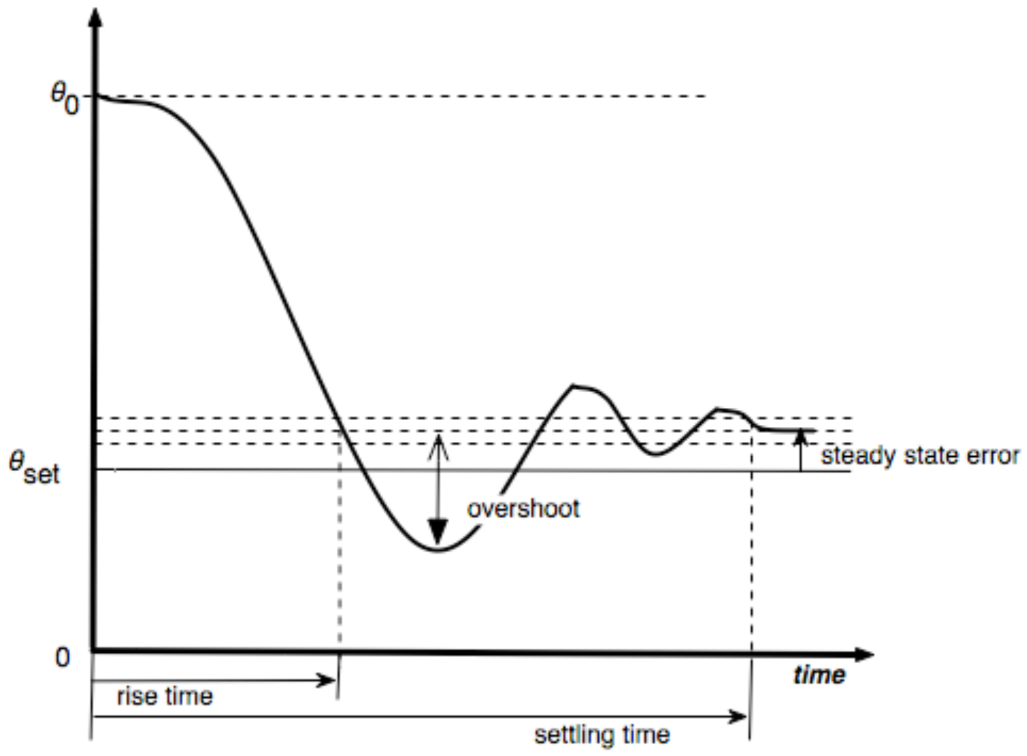

Figure 5.2 Transient response

Procedure

1.*Unit Step*

Set the temperature of the aluminum block to 30° C. Let the temperature stabilize then start your PWM with duty cycle set to 50%. Measure the transient response of the temperature controller (See the "Hints" section of the lab document on how to do this). Collect data till the response stabilizes. Use *Matlab* to graph temperature change vs time, your graph should resemble figure 5.3. From the numerical values or by observing the graph determine approximately the point of inflection for the response. Draw a tangent to the curve at the point of inflection (again approximately, you can use *Insert->Line* from the toolbar in *Matlab* to do this) and determine ***T*** and ***L*** as shown in Figure 5.3. Set the initial value of K_P as 0.9 $*$ (***T*** / ***L***) and K_I as 0.27 $*$ (***T*** / ***L***^{2}).

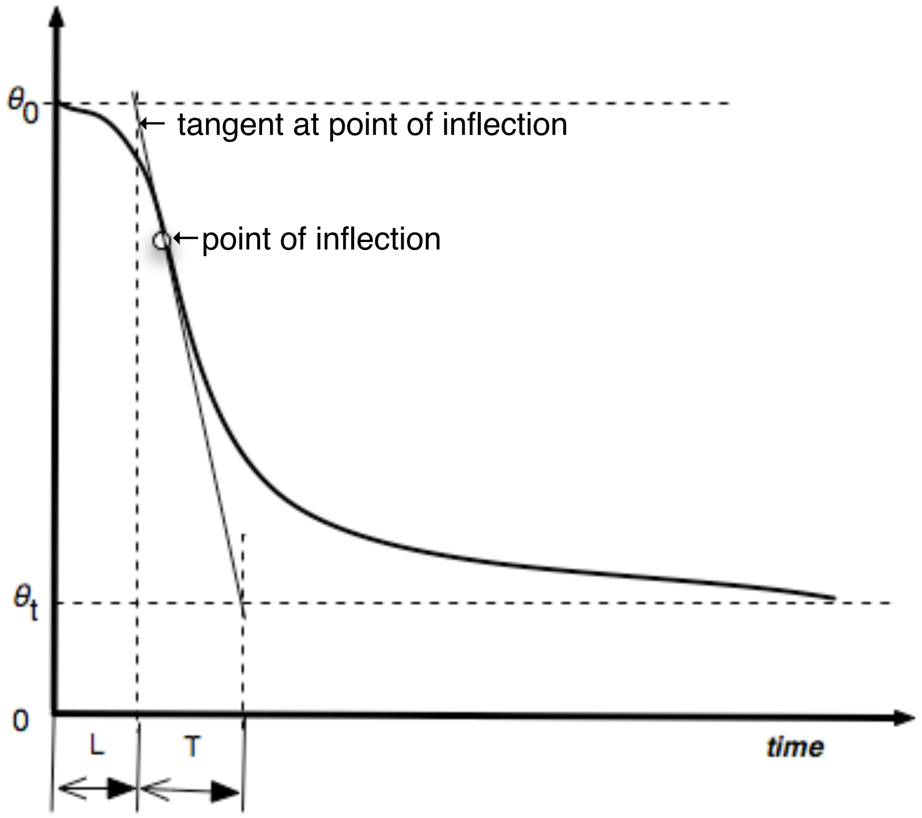

Figure 5.3 Transient response for unit step

2. *Tuning the proportional gain (K_P)*

The proportional term $K_P \bullet e(t)dt$ sets the system response directly proportional to difference (the error *e(t)*) between the measured temperature and the set point. Table 5.2 describes the effect of K_P on the system. To tune K_P to get the desired system response, start with K_I set to zero and with K_P initialized to the value computed in the unit step. Start your program and observe the response. If the rise-time is too slow increase K_P by factors of 10, if it is oscillating too much or the overshoot is large, decrease by factors of 10. Adjust K_P to the point where where the transient just starts to oscillate about the set point and then reduce its value gradually by factors of 2 till the response settles down with a reasonably small steady state error.

3. *Tuning the integral gain (K_I)*

The integral term $K_I \bullet \int e(t)dt$ provides a contribution proportional to the amount of time the error has been present and helps eliminate the steady state error. To tune K_I, initialize it to the value you determined in the unit step. With K_P fixed, vary the integral gain K_I gradually, using table 5.2 as a guide. You should be able to find a K_I that gives you the desired result without too much overshoot or oscillations.

PI controller response characteristics

The following table summarizes the effects of *increasing* the proportional and integral gains:

Gain	**Rise time**	**Overshoot**	**Settling time**	**Steady-state error**
K_P	decreases	increases	no significant effect	decreases
K_I	decreases	increases	increases	eliminated if tuned correctly

Table 5.2

References:

[1] Van de Vegte J., Feedback Control Systems (Prentice Hall Inc., 1994)

[2] http://www.embedded.com/2000/0010/0010feat3.htm