CHAPTER 4: MICROCONTROLLER BASED TEMPERATURE CONTROL SYSTEM

4.1 Introduction 148

4.2 Hardware Features 148
   4.2.1 Sensor 150
   4.2.2 Constant Current Source 151
   4.2.3 Instrumentation Amplifier 151
   4.2.4 Microcontroller 153
   4.2.5 Zero-Cross Detector 154
   4.2.6 Opto-Isolator 154
   4.2.7 Actuator 155

4.3 Software Features 160

4.4 Results and Discussions 172

References
4.1 Introduction

Temperature is one of the most widely measured and frequently controlled physical parameters in the industry. This is because quite often the processing and manufacturing of the desired product is possible only if the temperature is accurately measured and maintained. Further, it forms an important governing parameter in thermodynamic heat transfer and a number of chemical reactions/operations. The need for temperature control arises in various fields such as medical, biological, industrial and frequently in basic scientific research and R&D laboratories. Many physical and chemical reactions are sensitive to temperature and consequently, the temperature control is important in several industrial processes. Temperature control also finds application in cryostats that are used to perform experiments at very low temperatures in the field of spectroscopy, X-ray diffractometry and thermal microscopy.

Temperature control also plays a key role in many processes industries such as petrochemicals, thermal power stations, cement, paper-pulp, fertilizers etc; in addition, precision and quality in control of temperature is desirable to yield quality products. Hence, several investigators have designed and fabricated different types of temperature controllers [1-19]. But the attempts to use microcontroller with improved features like advanced SoC, microcontroller C8051F020 to control temperature are rather scarce in spite of several advantages that are associated with the use of such microcontrollers. Since C8051F020 is having all the features to work as a stand-alone system hence designing temperature control system with such microcontroller makes the system very compact, low power consumption and low-cost.

A microcontroller based precision temperature controller has been designed and fabricated, in order to control the temperature of the photoacoustic cell and to study the phase transitions of a sample. A PID control algorithm has been implemented for the present system, for controlling the temperature of photoacoustic cell at desired rate. This system consists of a temperature sensor (Pt-100), signal conditioner, data processing elements, display unit and microcontroller.

4.2 Hardware Features

Figure 4.1 shows the block diagram of C8051F020 microcontroller based temperature control system using PID controller. The system consists of the following elements:
Fig. 4.1 Block diagram of microcontroller based temperature control system
- Temperature sensor
- Constant current source
- Instrumentation amplifier
- Microcontroller board
- Zero-crossing detector
- Opto-isolator (DIAC)
- Actuator (TRIAC)

### 4.2.1 Temperature Sensor

The platinum resistance thermometer (Pt-100) is used for the present study to overcome the significant limitations of the conventional transducers such as non-linearity, low output, narrow range etc, Fig. 4.2 shows the photograph of Pt-100 which operates on the principle of change in electrical resistance of platinum wire as a function of temperature [20-21]. It is mechanically and electrically stable. One of the important features of Pt-100 is that the relation between temperature and resistance is linear, the drift error with ageing and handling errors are negligible. The relation between temperature and resistance of the platinum wire is given by the following relation.

\[ R_T = R_0 \left( 1 + AT + BT^2 + \ldots \right) \]

where all the second and higher order terms are negligibly small, and may be neglected.

The above equation is reduced to

\[ R_T \approx R_0 \left( 1 + AT \right) \]

----- (i)

Where \( R_T \) = Resistance at temperature \( T \)

\( R_0 \) = Resistance at temperature \( 0^\circ C \)

\( A \) = Co-eff of resistance = 0.00389 \( \Omega / ^\circ C \).

\( T \) = Temperature

![Fig. 4.2 PT-100 Platinum resistance thermometer](image-url)
The platinum resistance transducers are used for temperature measurement in the range of –220\(^0\)C to 750\(^0\)C and it has a temperature co-efficient of resistance as 0.00389 \(\Omega / ^0\)C.

### 4.2.2 Constant Current Source

The signal conditioning circuit is employed for converting resistance changes of the sensor into voltage changes. A constant current source shown in Fig. 4.3 has been designed using LM329 & operational amplifier [22]. This will eliminate the lead-wire resistance error of Pt-100. A stable voltage sources is constructed using LM329 (temperature compensated precision voltage source) which gives a constant voltage of 6.9V. This reference voltage is applied to the non-inverting input of op-amp (LM308). The Pt-100 acts as a feedback resistance and the resistance \(R_s\) determines the amount of current flowing through Pt-100. Since the potential difference between input terminals of the operational amplifier is zero. Hence, the voltage at the inverting terminal is also equal to 6.9V. Therefore the current flowing through the resistance \((R_s)\) is equal to 6.9V/\(R_s\). Since the input impedance of the amplifier is very high, the current flowing through Pt-100 is equal to the current flowing through resistance \(R_s\), thus \(R_s\) can be calculated by the following equation

\[
R_s = \frac{V}{I} = \frac{6.9V}{100\mu A} = 69.0k\Omega
\]

A 100k multi-turn potentiometer is used as \(R_s\) and it is adjusted to the value 69.0k, which gives a constant current of 100\(\mu\)A. The current flowing through Pt-100 should be as small as possible to avoid the self heating of the sensor.

### 4.2.3 Instrumentation Amplifier

When the current passes through the temperature sensor Pt-100, it produces a voltage given by the following relation

\[
\Delta V = 100\mu A \times (\text{Resistance of Pt-100})
\]

The differential voltage output of temperature sensor is very small, hence it has to be amplified so that further processing of signal can be made possible. This voltage is amplified through a instrumentation amplifier AD620 of Analog Device [23] make. The AD620 is a low cost, high accuracy instrumentation amplifier that requires only one external resistor to set gain of 1 to 10,000. The AD620, with its high accuracy of 40 ppm maximum nonlinearity, low offset voltage of 50 \(\mu\)Vmax, and offset drift of 0.6 \(\mu V/^0\)C.
max, is ideal for use in precision measurement systems. Fig. 4.4 shows the AD620 instrumentation amplifier designed with a gain of 100 to amplify the output of Pt-100.

![Fig. 4.3 Constant current source](image)

Further, more AD620 features 8-lead DIP packaging that is smaller than discrete designs. The AD620’s gain is varied by a single resistor \( R_G \), or more precisely, by whatever impedance appears between Pins 1 and 8. The AD620 is designed to offer accurate gains using 0.1% to 1% resistors.

For \( G = 1 \), the \( R_G \) pins are unconnected (\( R_G = \infty \)). For any arbitrary gain, \( R_G \) can be calculated by using the formula:

\[
R_G = 49.4k\Omega / (G-1)
\]

In the present application \( R_G \) is calculated for a gain of 100.

\[
R_G = 49.4k\Omega / (G-1) = 49.4k\Omega / (100-1) = 498 \Omega
\]

![Fig. 4.4 Instrumentation amplifier](image)
4.2.4 C8051F020 Microcontroller Board

The microcontroller used for the present study is C8051F020TB from Cygnal Integrated Products, Inc., Austin, USA. The photograph of C8051F020TB is shown in Fig. 4.5. The microcontroller board contains the salient features such as:

- High-speed pipelined 8051-compatible CIP-51 Microcontroller core (upto 25 MIPS)
- In-system, full-speed, non-intrusive debug interface (on-chip)
- True 12-bit (C8051F020) 100 KSPS, 8channel ADC with PGA and analog multiplexer
- True 8-bit DACs with programmable update scheduling
- 64k bytes of in-system programmable FLASH memory
- 4352 (4096+256) bytes of on chip RAM
- External data memory interface with 64k byte address space
- SPI, SM bus /I²C, and (2) UART serial interfaces implemented in hardware
- Five general-purpose 16-bit timers
- Programmable counter/Timer array with five capture / compare modules
- On-chip watchdog timer, VDD monitor, and temperature sensor
- 64-Digital port I/Os
- 2-Analog comparators

![Photograph of the C8051F020TB microcontroller board](image)

Fig. 4.5 Photograph of the C8051F020TB microcontroller board
4.2.5 Zero-Crossing Detector
The circuit diagram of zero crossing detector is presented in Fig. 4.6. The 12V transformer followed by the resistor and diodes produces a square wave at the input of an inverter A, which is synchronized with the AC line voltage but restricted to legal TTL levels. The RC and diode networks around the outputs of inverters A and B form differentiators. These differentiators produce positive and negative spikes at every zero crossing of the ac mains. The 7405 is an open collector hex inverter provided to assure adequate drive current for the differentiator. The negative spikes are clipped by the diodes D3 and D4 and the positive spikes are summed and inverted by inverter C, the output of the inverter is negative spikes synchronized with ac mains zero crossing.

At each zero crossing of the line, inverter C pulses and these pulses will interrupt microcontroller. At every interrupt the count from the output of PID is loaded into the on-chip timer-0 of the microcontroller. Then the counter decrements at every clock pulse until the count loaded in the timer-0 overflows at the rate decided by the on-chip clock i.e. 2MHz. This produces the pulse width modulated output to control the triac which in turn controls the power applied to the heater connected to the PA cell.

4.2.6 Opto-isolator
To isolate the AC power from DC powered boards, an opto-isolator MOC3011, of Motorola make as shown in Fig 4.7 is used. The MOC3011 contains LED and an opto-diac. The opto-diac conducts in both the directions of AC cycle and it triggers triac in both positive and negative cycle of ac signal in accordance with PWM signal generated through P6.0 port line as per the timer-0 count.
4.2.7 Actuator

An actuator is a final power control element which controls the power or energy supplied to the system to bring the physical parameter to the desired level. In the present study, as shown in Fig 4.7 a triac (BTA06) is used as final control element. Triac can conduct in both directions and is normally used in ac phase control. It can be considered as two SCRs connected in anti-parallel with a common gate connection. Since, triac is a bidirectional device its terminals cannot be designed as anode and cathode hence designated as MT1 and MT2. If terminal MT2 is positive with respect to terminal MT1, the triac can be turned ON by applying a positive gate signal between gate G and terminal MT1. It is not necessary to have both polarities of gate signal and triac be turned ON with either a positive or negative gate signal.

Here, the triac acts as a switch, when it is OFF, no power is allowed to pass through it to the load. When it goes ON, the load receives line voltage. This is quite adequate for simple ON or OFF operations. But to provide proportional control of power to the load, phase angle firing control technique has been employed. The firing angle is decided by the output of PID controller. With the help of zero cross detector the power applied to the heater (PA cell) is synchronized with line voltage, the timer is used to control the power based on the PID controller output.

Fig. 4.7 Opto- isolator and actuator
**Working of the System**

The photograph of the microcontroller based temperature control system designed and fabricated is as shown in Fig. 4.8 and the schematic diagram of microcontroller based temperature control system is shown in Fig. 4.9. The temperature of the photoacoustic cell is measured by using Pt-100 temperature sensor. Pt-100 produces change in resistance with change in temperature. This change in resistance is converted into change in voltage by using constant current source. The voltage corresponds to temperature is converted into digital data by the on-chip ADC of C8051F020 microcontroller. The digital data is converted into actual temperature by substituting it in the equation:

\[
\text{Temperature} = (2.657 \times \text{voltage}) - 273.0.
\]

The equation is derived from the temperature v/s output voltage graph of Pt-100 \(y = mx + C\). The present temperature of the photoacoustic cell is displayed on the LCD module. The microcontroller compares the present temperature with the set temperature and error is calculated. Then the error is applied to the PID equation (1).

\[
V_n = V_{n-1} + K_p(e_n - e_{n-1}) + K_i e_n + K_d(e_n - 2e_{n-1} + e_{n-2}) \quad \text{(1)}
\]

Where,

- \(V_n\) – Current output
- \(V_{n-1}\) – Previous output
- \(e_n\) – Error
- \(e_{n-1}\) – Previous error
- \(e_{n-2}\) – Previous to previous error
- \(K_p\) – Proportional Constant
- \(K_i\) – Integral constant
- \(K_d\) – Derivative constant

The control algorithm is implemented through embedded ‘C’ program. The output of the algorithm is fed into on-chip timer-0 of the microcontroller. The output from P6.0 port line of the microcontroller is connected to the opto-isolator (MOC 3011) which controls the firing angle of the triac. The firing angle is decided by the count in the timer-0 of the microcontroller which in-turn is proportional to the solution of PID equation (1). The usual method of controlling an alternating voltage (ac mains) is to vary the firing angle of triac. The small count in the timer-0 takes large time to overflow, this produces small
conduction angle (and hence small ON-time) and a small output power to the load, conversely a large count in the timer-0 will take less time for counter to overflow causing a large conduction angle (large ON time), hence, larger power is applied to the load. This variation in duty cycle controls the firing angle of the triac and hence an amount of energy supplied to the heater. Thus the later maintains the temperature of the photoacoustic cell at the desired value. The PID temperature controller designed is found to regulate the temperature of the photoacoustic cell within ±0.1°C. Fig. 4.10 shows the Photograph of microcontroller based temperature control system.

Fig. 4.8 PCB designed and fabricated for temperature control system
Fig. 4.9 Complete schematic of microcontroller based temperature control system
Fig. 4.10 Photograph of microcontroller based temperature control system
4.3 Software Features

The software for the present application is written in embedded ‘C’ program. The temperature control algorithms are written using Silicon Laboratories Integrated Development Environment (IDE) and Keil full-version embedded ‘C’ cross compiler.

4.3.1 Embedded ‘C’ program for temperature controller

The flowchart of C8051F020 microcontroller based temperature controller is shown in Fig. 4.11. The flowchart is self explanatory. The program first initializes the on-chip modules viz., ADC0, TIMER-0, UART0, Oscillator, ports and LCD module. At the beginning of the program the user should enter the maximum temperature and desired rate of increase of temperature. The PID constants are tuned in such a way that the rate of increase of temperature is 1°C/min. The program reads the voltage corresponding to the temperature through on-chip ADC0 and converts it into actual temperature by substituting in the equation derived by using least curve fitting method (y = mx+c). The measured temperature is compared with the set point and error is calculated. This error is applied to the PID controller. The PID controller produces the control action, hence the output of the algorithm is fed into on-chip timer-0. The on-chip timer is programmed in 16-bit timer mode whenever timer-0 overflows, the microcontroller sends high signal through P6.0 port line to final control element through opto-isolator. The program sends the temperature information to C8051F020 microcontroller through UART0 and displays the same on LCD module. The above procedure is repeated continuously till it attains the desired set point temperature. Again set point will be incremented by 1°C. The above steps are repeated to get desired ramp type linearly increasing temperature up to maximum temperature entered at the beginning of the program.
Initialize on-chip peripherals viz., ADC0, TIMER-0, UART0, Oscillator, Ports and LCD module

Set the maximum temperature (set point) and rate of increase of temperature to study the sample characteristics as a function of temperature

Initialize variables and PID constants
\( K_p = 200, K_i = 4.0, K_d = 0.5; \)
\( v_n, e_n, v_{n-1} = 0.0, e_{n-1} = 0.0, e_{n-2} = 0.0, u_{k1}=0.0 \)

Read the voltage from the Pt-100 signal conditioner output through on-chip A/D converter of C8051F020 microcontroller

Calculate the Temperature by substituting this voltage into the equation
\( Y = mx+C \)

Display measured temperature on LCD module

Send the measured temperature to the Personal Computer through the UART-0

Compute the error
\( e_n = \text{set point} - \text{measured value} \)

Apply error to the PID algorithm
\( v_n = v_{n-1} + K_p(e_n - e_{n-1}) + K_i e_n + K_d(e_n - 2e_{n-1} + e_{n-2}); \)

**Fig. 4.11 Flowchart of temperature control system (Continued…)**
Apply output of PID to the on-chip Timer-0

Update PID parameters
vn_1 = vn;
en_2 = en_1;
en_1 = en

Is set point attained?

Increase the set point by 1°C

Is maximum temperature attained?

End
Embedded ‘C’ program for temperature control system

// ------------------
// Includes
// ------------------
#include <c8051f020.h> // SFR declarations
#include <stdio.h>
// ------------------
// 16-bit SFR Definitions for F02x
// ------------------
sfr16 ADC0 = 0xbe; // ADC0 data
sfr16 RCAP2 = 0xca; // Timer2 capture/reload
sfr16 RCAP3 = 0x92; // Timer3 capture/reload
sfr16 TMR2 = 0xcc; // Timer2
sfr16 TMR3 = 0x94; // Timer3
// ------------------
// Global Constants
// ------------------
#define BAUDRATE 9600 // Baud rate of UART in bps
#define SYSCLK 22118400 // External crystal oscillator frequency
// #define SAR_CLK 2500000 // Desired SAR clock speed
// ------------------
// Function Prototypes
// ------------------
void OSCILLATOR_Init(void);
void PORT_Init(void);
void UART0_Init(void);
void ADC0_Init(void);
void LCD_init(void);
void LCD_cmd(void);
void LCD_data(void);
void Delay(void);
void TEMP_Cal(void);
void LCD_MS_Disp(void);
void LCD_SP_Disp(void);
void Temp_display(void);
void Timer0_Init(void);
// ------------------
// main() Routine
// ------------------
void main (void)

float vn,en,vn_1 = 0.0, en_1 = 0.0,en_2 = 0.0;
long uk,uk1;
void main (void)
{ 
    WDTCN = 0xde;                          // Disable watchdog timer
    WDTCN = 0xad;
    OSCILLATOR_Init();
    XBR0  = 0x04;                       // Enable UART0  Tx = P0.0; Rx = P0.1
    XBR1  = 0x04;     // Enable INT0 to P0.2
    XBR2  = 0x40;                       // Enable crossbar and weak pull-up
    P0MDOUT |= 0x01;                       // Set TX pin to push-pull
    P74OUT |= 0xFF;
    UART0_Init ();                         // Initialize UART1
    ADC0_Init();                           // Init ADC
    LCD_init();
    Timer0_Init();
    while(1)

    TEMP_Cal();                              // Call sub-routine to measure temperature
    SET_TEMP = Temp;
    LCD_SP_Disp();                           // Call sub-routine to display set point temperature
    P4 = 0xc0;
    LCD_cmd();
    LCD_MS_Disp();                           // Call sub-routine to display measured temperature
    Delay();
    Temp = Temp*100;
    // Separate the digits to display temperature
    lb = (int)Temp%100;
    hb = (int)Temp/100;
    while(TI0 != 1);
    TI0 = 0;
    SBUF0 = hb;
    while(TI0 != 1);
    TI0 = 0;
    SBUF0 = lb;

    rep1:  P4=0x02;
        LCD_cmd();
        TEMP_Cal();
        LCD_SP_Disp();
        P4 = 0xc0;
        LCD_cmd();
        LCD_MS_Disp();
    en = SET_TEMP - Temp;
    vn = vn_1 + Kp*(en - en_1)+ Ki*en + Kd*(en - 2*en_1 + en_2);   // PID equation implementation through difference equation
    uk = (int)vn;
    uk1 = 47536+uk;
}
vn_1 = vn;
en_2 = en_1;
en_1 = en;
Temp = Temp*100;
Temp_int = (int) Temp;

lb = Temp_int%100;
hb = Temp_int/100;

while(TI0 != 1);
TI0 = 0;
SBUF0 = hb;

while(TI0 != 1);
TI0 = 0;
SBUF0 = lb;
EA = 1;
for (s=1;s<3000;s++)
for (h=1:h<3000;h++)
if(en<=0)
SET_TEMP = SET_TEMP + 1.0;  //increment step by 1 °C
//if temperature of PA cell reached
//maximum value then goto end else
//repeat above steps

goto rep2;
goto rep1;
rep2:
TMOD = 0x01;
TL0 = 0xff;
TH0 = 0xb8;
EA = 0;
while(1);

void Timer0_int(void) interrupt 1  // timer-1 interrupt subroutine to
// trigger triac
{
    EA = 0;
    ETO = 0;
    TR0 = 0;
    IE0 = 0;
    P6 = 0x01;
    Delay();
    P6 = 0x00;
    EX0 = 1;
    EA = 1;
}
void Int0_int(void) interrupt 0 // timer-0 interrupt to load output of PID equation onto timer-0 for getting desired firing angle.
{
  EA = 0;
  EX0 = 0;
  TMOD = 0x01;
  if (uk1 >= 47536 && uk1 <= 63740) // check whether control action (count) lies between 0-10msec, then load control action onto timer-0
  {
    TL0 = uk1 % 256;
    TH0 = uk1 /256;
  }
  else if(uk1 < 47536) // else load timer-0 with B8FF for applying minimum voltage (3.1V)
  {
    TL0 = 0xff;
    TH0 = 0xb8;
  }
  else // else load timer-0 with F8FF to get maximum voltage (220V)
  {
    TL0 = 0xff;
    TH0 = 0xf8;
  }
  TR0 = 1;
  ET0 = 1;
  EA  = 1;
}
void LCD_SP_Disp(void) //Display set temperature
{
  int i;
  for(i=0;i<9;i++)
  {
    P4 = LCDSP[i];
    LCD_data();
  }
  d= SET_TEMP*100;
  d = (int)d;
  d1 = d/10000;
  if(d1 != 0)
  {
    d1 = d1 + 0x30;
    P4 = d1;
    LCD_data();
  }
  d2 = d%10000;
  d2 = d2/1000;
  d2 = d2 + 0x30;
void LCD_MS_Disp(void) //Display Subroutine to display measured temperature
{
    int i;
    for(i=0;i<9;i++)
    {
        P4 = LCDMS[i];
        LCD_data();
    }
    if (Temp<0)
    {
        Temp = -Temp;
        P4 = '-';
        LCD_data();
    }
    d = Temp*100;
    d = (int)d;
    d1 = d/10000;
    if(d1 != 0)
    {
        d1 = d1 + 0x30;
        P4 = d1;
        LCD_data();
    }
}
d2 = d % 10000;
d2 = d2 / 1000;
d2 = d2 + 0x30;
P4 = d2;

LCD_data();
d3 = d % 1000;
d3 = d3 / 100;
d3 = d3 + 0x30;
P4 = d3;
LCD_data();
P4 = '\';
LCD_data();
d4 = d % 100;
d4 = d4 / 10;

d4 = d4 + 0x30;
P4 = d4;
LCD_data();
P4 = 0xdf;
LCD_data();
P4 = 'C';
LCD_data();
Delay();
}

//---------------------------------------------
// LCD_Init Subroutine
//---------------------------------------------
void LCD_init(void)
{
    P4 = 0x38;
    LCD_cmd();
P4 = 0x0e;
    LCD_cmd();
P4 = 0x06;
    LCD_cmd();
P4 = 0x01;
    LCD_cmd();
P4 = 0x02;
    LCD_cmd();
}

//---------------------------------------------
// LCD_cmd Subroutine
//---------------------------------------------
void LCD_cmd(void)
{
    int n;
    P5=0x02;
    for(n=1;n<1000;n++);
    P5=0x00;
    for(n=1;n<1000;n++);
}

//------------------------------------------------------------------------------------
// LCD_data Initialization Subroutine
//------------------------------------------------------------------------------------

void LCD_data(void)
{
    int m;
    P5=0x03;
    for(m=1;m<1000;m++);
    P5=0x01;
    for(m=1;m<1000;m++);
}

void TEMP_Cal(void)
{
    int k1;
    sum=0.0;
    for(k1=1;k1<=1000;k1++)
    {
        AD0INT=0;
        AD0BUSY=1;
        while(AD0INT!=1);
        hb=ADC0H & 0x0F;
        lb=ADC0L;
        v1=(hb*256)+lb;
        v2=(v1*2.445)/4096.0;
        sum=sum+v2;
    }
    v2=sum/1000.0;
    Temp = ((2.657*v2*1000)+ 5.01)-273.0;
}

void UART0_Init (void)
{
    CKCON = 0x20;  // Timer 2 uses the system clock
    T2CON = 0x34;  // Timer 2 used for TX and RX, enabled
RCAP2 = -(long) (SYSCLK/BAUDRATE)/32;
TMR2 = RCAP2;
TR2= 1;                                // Start Timer2

SCON0 = 0x50;                          // 8-bit variable baud rate;
// 9th bit ignored; RX enabled
// clear all flags
TI0     = 1;                            // Indicate TX0 ready

void Timer0_Init (void)
{
    TMOD   = 0x01;                        // TMOD: timer 0, mode 1, 16-bit
    CKCON |= 0x00;                         // Timer0 uses SYSCLK /12 as time
    // base
    TCON = 0x01;     // Transition actived
    IE = 0x03;      // Global enable EA, Timer0 and
    // INTO enable
    // PCON |= 0x80;                         // SMOD00 = 1
    // TI0    = 1;                             // Indicate TX0 ready

    ADC0_Init Subroutine

    ADC0_Init (void)
    {
        AMX0CF    = 0x00;    // AIN inputs are single-ended
                                // (default)
        ADC0CN    = 0x00;       // Enable temp sensor, on-chip
                                // VREF,
        REF0CN    = 0x03;             // turn on bias generator and
                                // internal reference.
        AMX0SL    = 0x00;      // Single-ended mode
        ADC0CF = (SYSCLK/SAR_CLK) << 3;
        ADC0CF = 0x40;       // PGA gain = 1 (default)
        AD0EN   = 1;
    }

    TIMER3_Init

    TIMER3_Init (int counts)
    {
        TMR3CN = 0x02;                // Stop Timer3; Clear TF3; set sysclk
                                // as timebase
        RCAP3 = -counts;              // Init reload values
        TMR3  = RCAP3;                // Set to reload immediately
        EIE2 &= ~0x01;                // Disable Timer3 interrupts
        TMR3CN |= 0x04;               // start Timer3
    }
/* 
// This function initializes the system clock to use the external 22.1184MHz crystal.
//-----------------------------------------------------------------------------
void OSCILLATOR_Init (void)
{
    Int i;                                  // Software timer
    OSCICN |= 0x80;                        // Enable the missing clock detector
    // Initialize external crystal oscillator to use 22.1184 MHz crystal
    OSCXCN = 0x67;                         // Enable external crystal osc.
    for (i=0; i < 256; i++);               // Wait at least 1ms
    while (!(OSCXCN & 0x80));             // Wait for crystal osc to settle
    OSCICN |= 0x08;                     // Select external clock source
    OSCICN &= ~0x04;                    // Disable the internal osc.
}
void Delay(void)
{
    int i,k:
    for(i=0;i<5;i++)
        for(k=0;k<10;k++)
    }
//----------------------------------------------------------------------------
// End Of File
//----------------------------------------------------------------------------
4.4 Result and Discussion

The PID controller is implemented to control the temperature of the photoacoustic (PA) cell to study the amplitude and phase variations in the samples. A ramp input is applied to the PA cell where the temperature is varied from initial room temperature of 27.0 °C to final temperature of 200.0 °C with a slope of 0.1°C/Min. The PID response with respect to ramp input is shown in the Fig. 4.12. It is quite evident from the graph that, the controller exhibit good tracking response with linear characteristics.

![PID controller response for ramp input](image)

**Fig. 4.12 PID controller response for ramp input**

In the present work C8051F020 microcontroller based temperature control system for the photoacoustic studies is designed and fabricated. The presently designed temperature control system exhibits good tracking response their by precise control of temperature is achieved. The temperature is varied at the rate of 0.1°C/Min and can be employed for photoacoustic spectrometer applications.
REFERENCES


14. Wei Li; Chao Dai; Lei Liu; Yalou Li “Optimization Model and PID Temperature Control System Design for CO2 Capture Process by CaO


24. Aidan O’Dwyer - Handbook of PI and PID Controller Tuning Rules

25. 2009 - Technology & Engineering