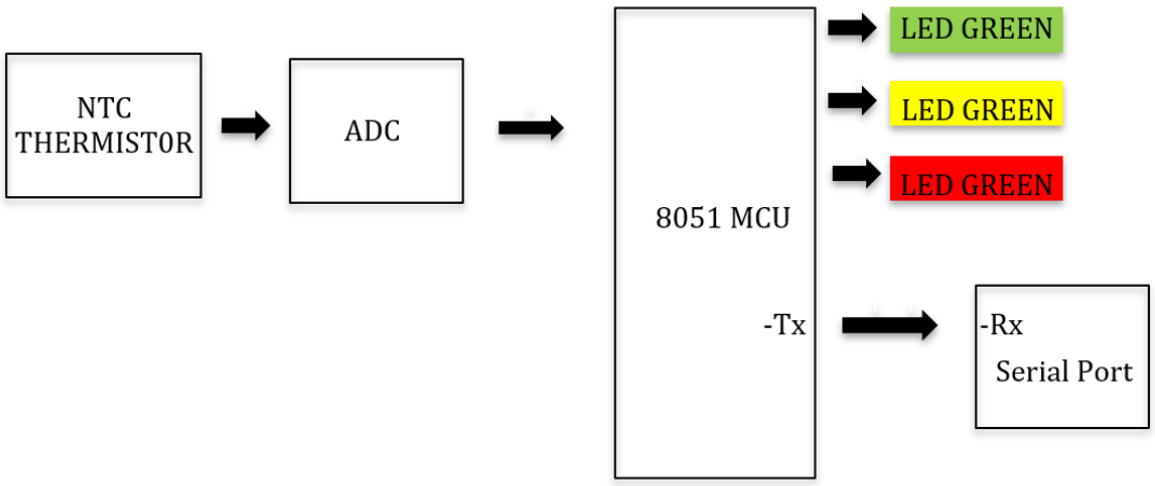
Interfacing a generic 10K NTC Thermistor to 8051MCU

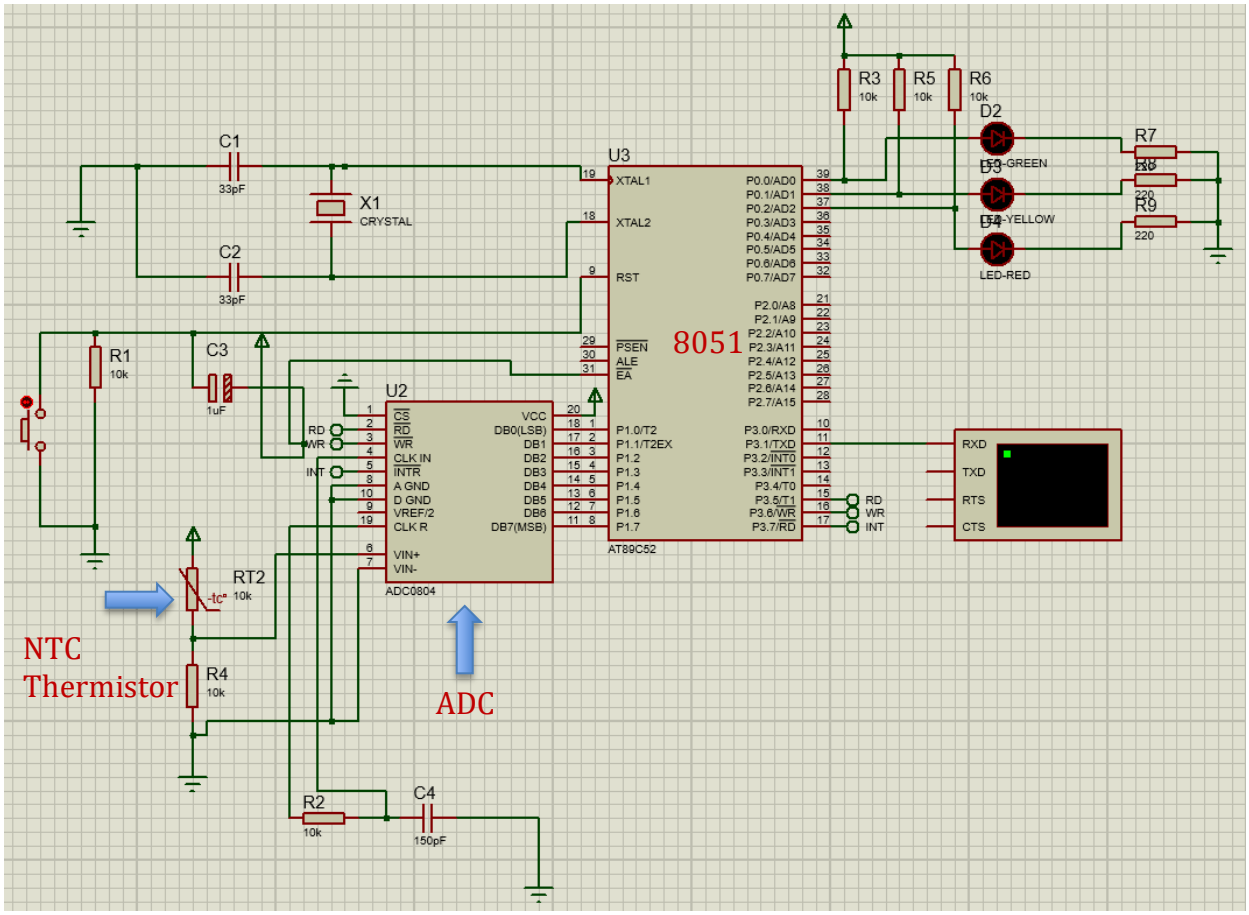
Introduction

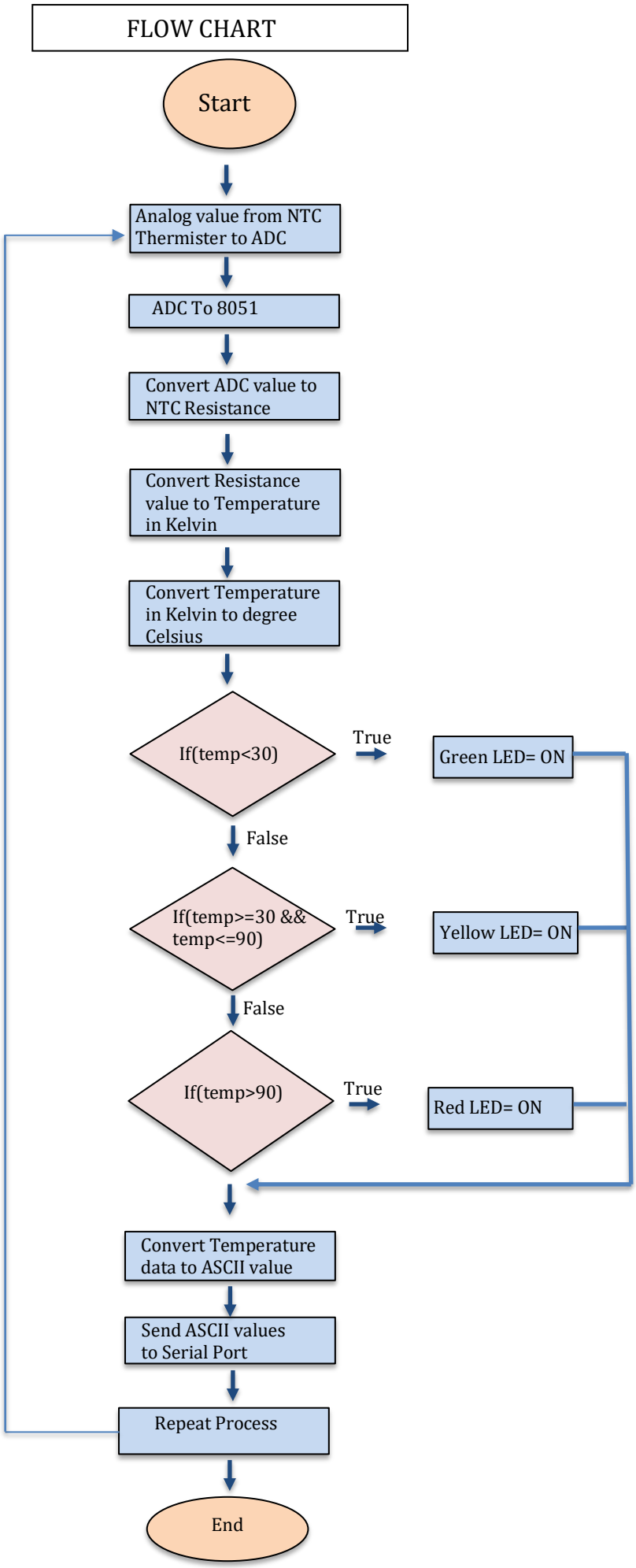
- A thermistor is a variable resistance element, whose resistance changes with the change in temperature.
- The resistance decreases with an increase in temperature for NTC (Negative Temperature Coefficient) type thermistors.
- The change in resistance value is a measure of the temperature.
- By using the thermistor in series with a fixed resistance forming a simple voltage divider network, we can find out the temperature by the change in voltage of the voltage divider network due to a change in temperature. (Temperature change leads to change in resistance which leads to change in voltage.)

Block Diagram



Circuit Diagram

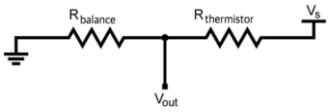



FLOW CHART
Start
Analog value from NTC Thermister to ADC
ADC To 8051
Convert ADC value to NTC Resistance
Convert Resistance value to Temperature in Kelvin
Convert Temperature in Kelvin to degree Celsius
If(temp<30)
True
Green LED= ON
False
If(temp>=30 && temp<=90)
True
Yellow LED= ON
False
If(temp>90)
True
Red LED= ON
Convert Temperature data to ASCII value
Send ASCII values to Serial Port
Repeat Process
End

Working Process

Step 1: Getting Data from sensor

We are using temperature sensor NTC thermistor 10k resistance at 25 degree Celsius. Resistance of thermistor is decreased with temperature. We make voltage divider circuit by connecting 10k resistor series to it to find vout of thermistor thus finding internal resistance of NTC thermistor. The output of this circuit is connected to Analog to digital convertor ADC0804.



$$V_{out} = V_s \cdot \left(\frac{R_{balance}}{R_{thermistor} + R_{balance}}\right) \quad R_{thermistor} = R_{balance} \cdot \left(\frac{V_s}{V_{out}} - 1\right)$$

Step 2: Reading ADC Values

The output ADC is connected to port P1 of 8051. Using while loop we get input data continuously The ADC value is used to find out variable resistance of thermistor and temperature.

Step 3: Finding temperature from ADC Value

We need to find temperature using Steinhart equation given below. The output will be in farenhit scale we further subtract this with 273.5 to get degree celsius.

$$\frac{1}{T} = \frac{1}{T_o} + \left(\frac{1}{\beta}\right) \cdot \ln\left(\frac{R}{R_o}\right)$$

T =Temperature
Beta =3950 (as per datasheet)
To = Room temperature
R =Thermistor temperature
Ro = Resistance at reference temperature T_o.

Step4: Sending data to serial port.

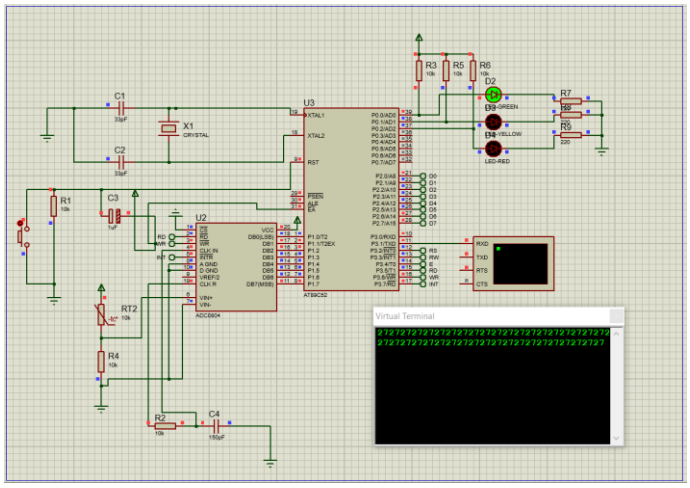
In order to print values of temperature we send temperature values to serial port via txd of 8051 after converting to ASCII values.

Step5: Controlling LED's

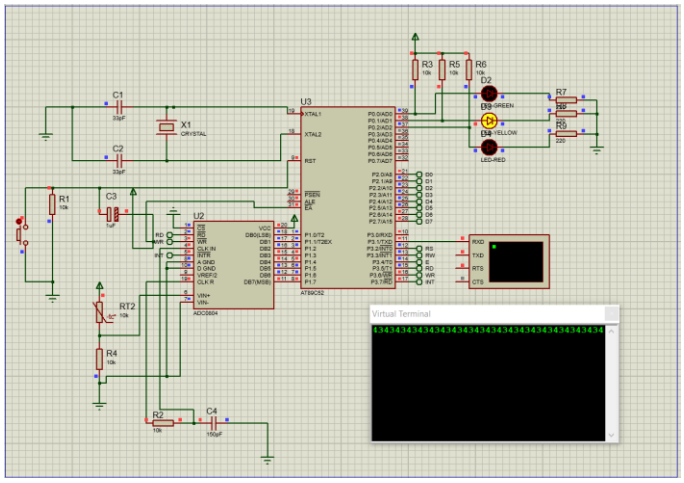
We continuously check the conditions statements for temperature values through if else condition statements. If temperature is less than 30 degree Celsius green LED is ON, if the temperature is greater then 90 degree Celsius red LED is ON and if temperature is between 30 degree Celsius and 90 degree Celsius yellow LED is ON.

OutPut:

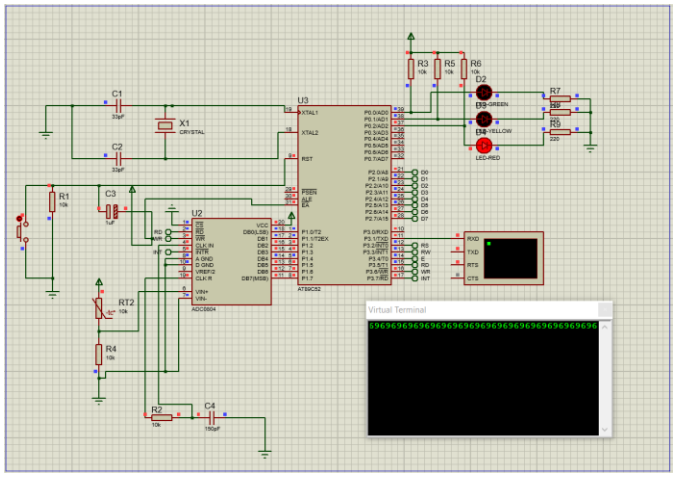
1)Temperature less than 30 degree Celsius



2)Temperature between 30 degree Celsius and 90 degree Celsius.



3)Temperature greater then 90 degree Celsius.



C Program

```
#include<reg51.h>
#include<math.h>
sbit LED_GREEN=P0^0;
sbit LED_YELLOW=P0^1;
sbit LED_RED=P0^2;
sbit wr=P3^6;
sbit rd=P3^5;
sbit intr=P3^7;

sbit rs=P3^2;
sbit rw=P3^3;
sbit en=P3^4;

const double Rb          = 9710.0;
const double ADC_max     = 255.0;
const double BETA        = 3950.0;
const double Room_Temp    = 298.15;   // room temperature in Kelvin
const double RES_Room_Temp = 10000.0;
void ser(unsigned char data1);
void conv_ASCII(unsigned char value);
int adc();
double temperature(unsigned char val);
void delay(unsigned int v);
void main()
{
                        LED_GREEN=0;
                        LED_YELLOW=0;
                        LED_RED=0;
                        intr=1;
                        P1=0x0FF;
                        TMOD=0x20;
                        SCON=0x50;
                        TH1=0xFD;
                        TR1=1;
                        while(1)
                        {
                         unsigned char tCelsius;
                          tCelsius=adc();
                         if(tCelsius<30)
                        {
                         LED_GREEN=1;
                          LED_YELLOW=0;
                          LED_RED=0;
                        }
                        else if(tCelsius>=30 && tCelsius<=90)
                        {
                         LED_YELLOW=1;
                         LED_GREEN=0;
                          LED_RED=0;
                        }
                        else if(tCelsius>90)
                        {
                         LED_RED=1;
                         LED_GREEN=0;
                          LED_YELLOW=0;
                        }
                         delay(150);
                        }
}
```

```
//ANALOG TO DIGITAL CONVERSION
int adc()
{
                        char adc_val,temp_cel;
                        rd=1;
                        wr=0;
                        delay(50);
                        wr=1;
                        while(intr==1);
                        rd=0;
                        adc_val=P1;
                        temp_cel=temperature(adc_val);
                        conv_ASCII(temp_cel);
                        return temp_cel;

}

double temperature(unsigned char val)
{
 // variables that live in this function
 double RNtc = 0;          // Holds thermistor resistance value
 double tK   = 0;          // Holds calculated temperature
 double tC   = 0;          // Hold temperature in celsius

 RNtc = Rb * ( (ADC_max / val) - 1);

 tK = (BETA * Room_Temp) /
       (BETA + (Room_Temp * log(RNtc / RES_Room_Temp)));


 tC = tK - 273.15;  // convert kelvin to celsius

 return tC;    // Return the temperature in Celsius
}

//CONVERT TO ASCII
void conv_ASCII(unsigned char value)
{
 unsigned char x1,x2,x3;
 x1=(value/10);
 x1=x1+(0x30);
 x2=value%10;
 x2=x2+(0x30);
 x3=0xDF;
 ser(x1);
 ser(x2);
}
//SERIAL COMMUNICATION
void ser(unsigned char data1)
{
                        SBUF=data1;
                        while(TI==0);
                        TI=0;
}
void delay(unsigned int v)
{
                        unsigned char i;
                        for(i=0;i<v;i++);
}
```