8051 MICROCONTROLLER

Trainer : Dr. Jeevan K M

INTRODUCTION

Embedded system Definition : An embedded system is a combination of computer hardware and software designed for a specific function.

Why is 8051 called an 8-bit microcontroller ?

The 8051 microcontroller is called an 8-bit microcontroller because it processes data in 8-bit chunks. It has an 8-bit data bus, 8-bit wide registers, and an instruction set designed for 8-bit operations. The internal RAM and special function registers are also 8 bits wide. This 8-bit architecture defines its processing capabilities.

1.LED Blinking Using 8051 Microcontroller

Introduction

LED blinking is a fundamental experiment to understand the working of microcontrollers like the 8051. By controlling the LEDs using ports of the microcontroller, various blinking patterns can be achieved. This report outlines the necessary setup and methods to implement LED blinking patterns using an 8051 microcontroller.

Setup and Configuration

1. Microcontroller and LED Connection:

    - LEDs are connected to the microcontroller's port (e.g., Port 1, denoted as P1).
    - Each LED is connected to a specific pin on the port (P1.0, P1.1, ..., P1.7).
    - The cathode of each LED is connected to the ground (GND).
    - The anode of each LED is connected to the respective port pin through a current-limiting resistor.
2. Port Assignment:
    - By assigning a hexadecimal value to the port (e.g., P1 = 0x01), the microcontroller converts this value to binary.
    - Each bit in the binary representation controls an individual LED:
        - A bit value of '1' turns the corresponding LED on.
        - A bit value of '0' turns the corresponding LED off.
    - For example, if P1 = 0x01 (binary 0000 0001), only the LED connected to the least significant bit (P1.0) will be on, and the rest will be off.

Circuit Diagram

- Anode of each LED -> Connected to a specific port pin (P1.0 to P1.7)
- Cathode of each LED -> Connected to Ground (GND) through a resistor.

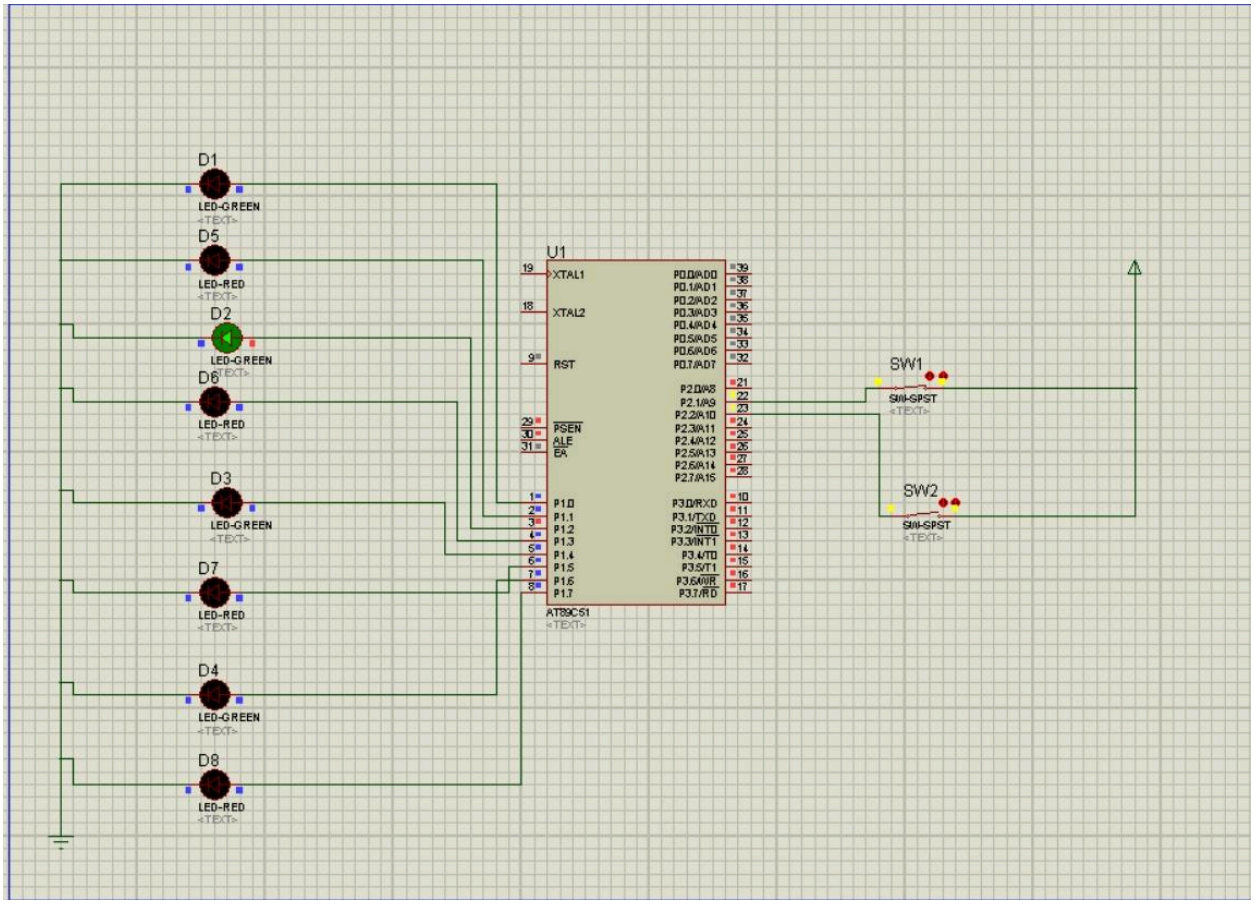
CASE 1 : Considering 8 Led's,which should glow alternatively

Code:

cs1.c | Blink_LED.c | led_bilnkcase2.c | LCD.c | led_day3.c

```c
#include<reg51.h>
sbit sw1 = P2^1;
sbit sw2 = P2^2;
void delay(unsigned int);
void main(void)
{
  sw1=0;       //0 voltage for logic0 , 1 for logic1
  sw2=0;
  while(1)    //infinite loop
  {
    if(sw1==0 && sw2==0)
    {
      P1=0x00;
    }
    else if(sw1==0 && sw2==1)
    {
      P1=0xAA;
    }
    else if(sw1==1 && sw2==0)
    {
      P1=0x55;
    }
    else if(sw1==1 && sw2==1)
    {
    //while(1)  //repeat forever
      //{
        P1=0x55;
        delay(500);
        P1=0xAA;
        delay(500);
      //}
      }
  }
}
//void delay(void)
//{
  //TMOD=0X01;
  //TL0=0XC0;
  //TH0=0X63;
  //TR0=1
//  WHILE(TF==0);
  //TR0=0;
  //TF==0;
 //}
void delay (unsigned int t)
{
  unsigned int i,j;
  for(i=0;i<t;i++)
  for(j=0;j<1275;j++);
}
```

PROTEUS SIMULATION :



CASE 2 : Considering 8 Led's, and two switches . When switch 1 is closed the last 4 leds glow and , When switch 2 is closed the first 4 leds glow.

Summary:

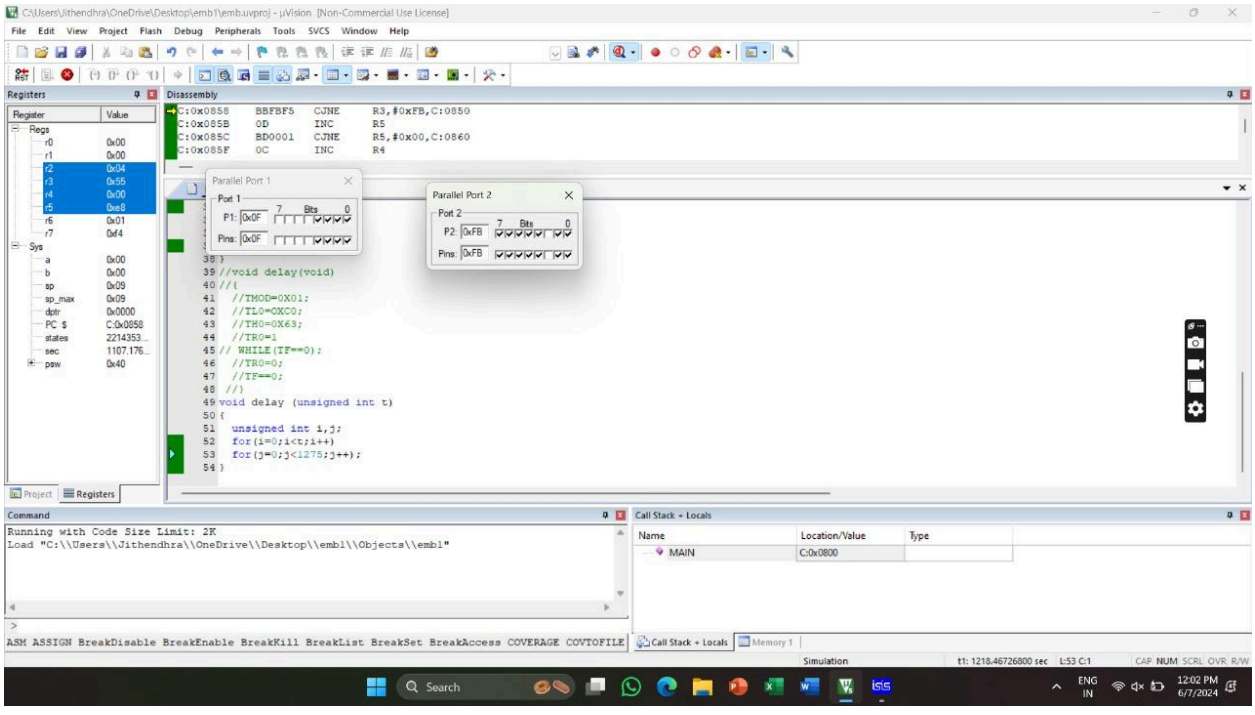
Closing Switch 1 -> LEDs 5, 6, 7, and 8 will light up.

Closing Switch 2 -> LEDs 1, 2, 3, and 4 will light up.

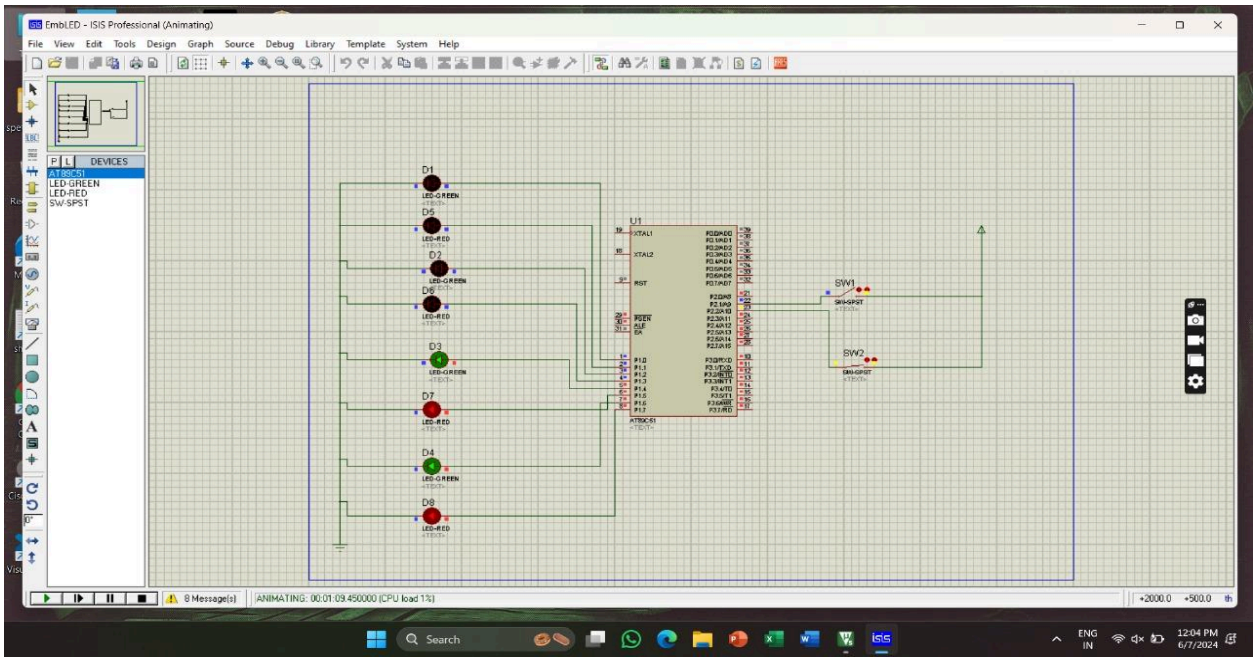
CODE:

cs1.c | Blink_LED.c | led_bilnkcase2.c | LCD.c | led_day3.c

```
#include<reg51.h>
sbit sw1 = P2^1;
sbit sw2 = P2^2;
void delay(unsigned int);
void main(void)
{
   sw1=0;       //0 voltage for logic0 , 1 for logic1
   sw2=0;
   while(1)    //infinite loop
   {
     if(sw1==0 && sw2==0)
     {
       P1=0x00;
     }
     else if(sw1==0 && sw2==1)
     {
       P1=0xF0;
       delay(500);
       P1=0X00;
     }
     else if(sw1==1 && sw2==0)
     {
       P1=0X0F;
       delay(500);
       P1=0X00;
     }
     else if(sw1==1 && sw2==1)
     {
     //while(1)  //repeat forever
       //{
         P1=0x55;
         delay(500);
         P1=0xAA;
         delay(500);
       //}
       }
   }
}
//void delay(void)
//{
```



Proteus Simulation :

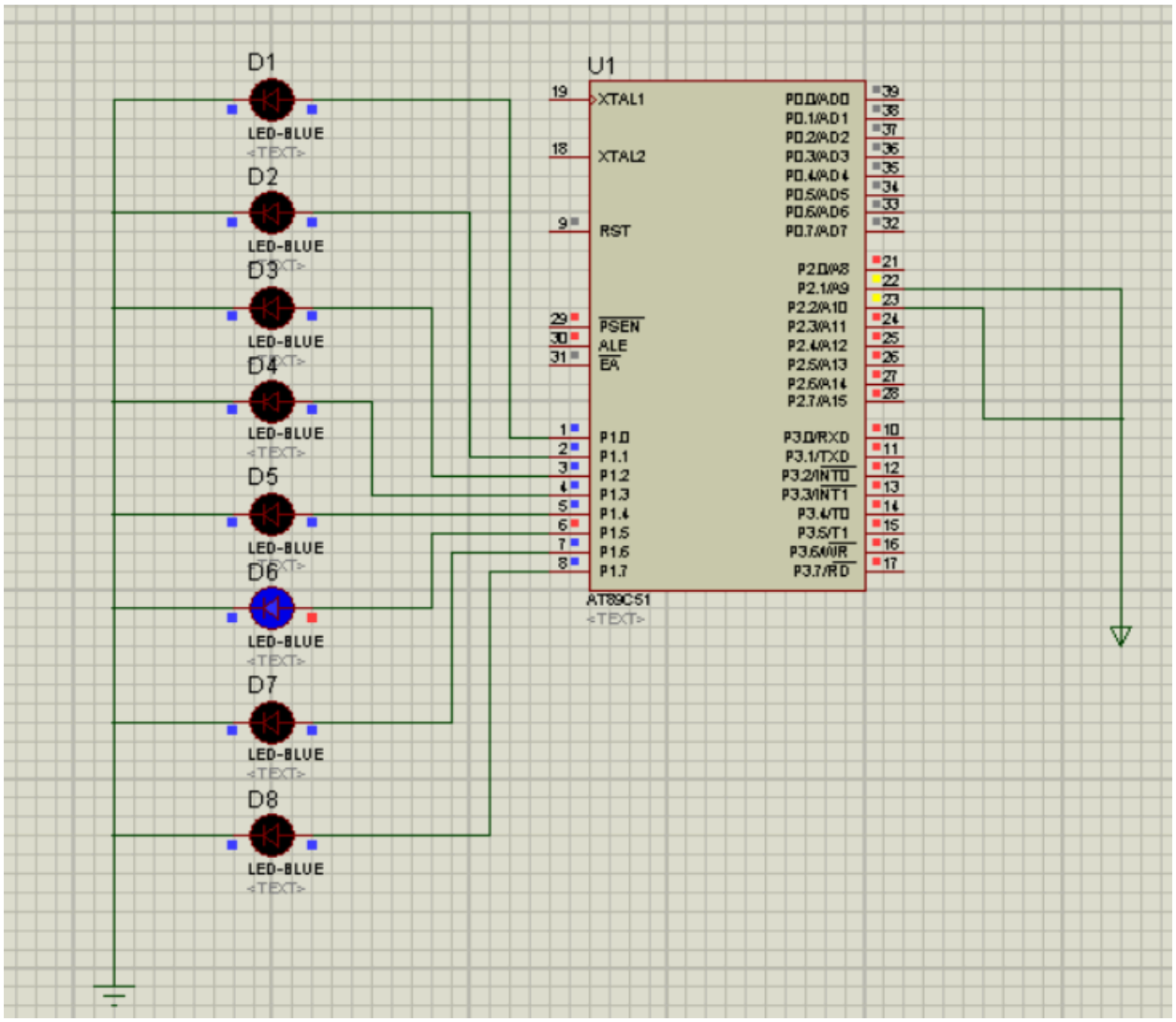


CASE 3 : Considering 8 Led's , all leds have to blink with a certain delay.

CODE :

```
#include <reg51.h>
sbit sw1 = P2^1;
sbit sw2 = P2^2;

void delay(unsigned int);
void main(void)
{
   sw1 = 0;
   sw2 = 0;
   while(1)
     {
     P1=0X80;
     delay(500);
     P1=0X40;
     delay(500);
     P1=0X20;
     delay(500);
     P1=0X10;
     delay(500);
     P1=0X08;
     delay(500);
     P1=0X04;
     delay(500);
     P1=0X02;
     delay(500);
     P1=0x01;
     delay(500);
     }
}
     void delay(unsigned int t)
{
   unsigned int i,j;
   for(i=0;i<t;i++)
   for(j=0;j<1275;j++); // For 1ms, for loop need to iterate 1275 times.
}
```

PROTEUS Simulation :



2.ADC Using 8051 Microcontroller :

Learning Objective: Understanding the process of converting an analog signal to a digital signal using the ADC 0808. Learning how to interface the ADC 0808 with the 8051 microcontroller.

Inputs and Outputs

Input: A potentiometer (1k or 10k ohms).

Outputs:

- LEDs.
- Logic
- Analog Input: The analog output from the potentiometer is fed into one of the input channels of the ADC 0808.

ADC to Microcontroller Interface:

- The digital output pins (D0-D7) of the ADC 0808 are connected to Port 1 of the 8051 microcontroller.
- The selection lines (A, B, C) of the ADC are connected to Port 2 of the 8051 to select the input channel.
- Additional control lines such as End of Conversion (EOC), Start, Address Latch Enable (ALE), and Clock Pulse are also connected to Port 2.
- Digital Output:
- The LEDs, which display the digital value of the potentiometer reading, are connected to Port 3 of the 8051 microcontroller.

- Based on the variation in resistance of the potentiometer, the digital value output from the ADC will change, which in turn will change the state of the LEDs connected to Port 3.

Circuit Diagram

Potentiometer:

- Connect the wiper of the potentiometer to the analog input channel (e.g., IN0) of the ADC 0808.
- Connect the other two terminals of the potentiometer to Vcc and GND.
- ADC 0808 to 8051 Microcontroller:
- Digital Output Pins (D0-D7): Connect to Port 1 (P1.0 to P1.7) of the 8051.
- Selection Lines (A, B, C): Connect to Port 2 pins (e.g., P2.0, P2.1, P2.2).

Control Lines:

- EOC: Connect to a pin on Port 2 (e.g., P2.3).
- Start: Connect to a pin on Port 2 (e.g., P2.4).
- ALE: Connect to a pin on Port 2 (e.g., P2.5).
- Clock Pulse: Connect to a pin on Port 2 (e.g., P2.6).

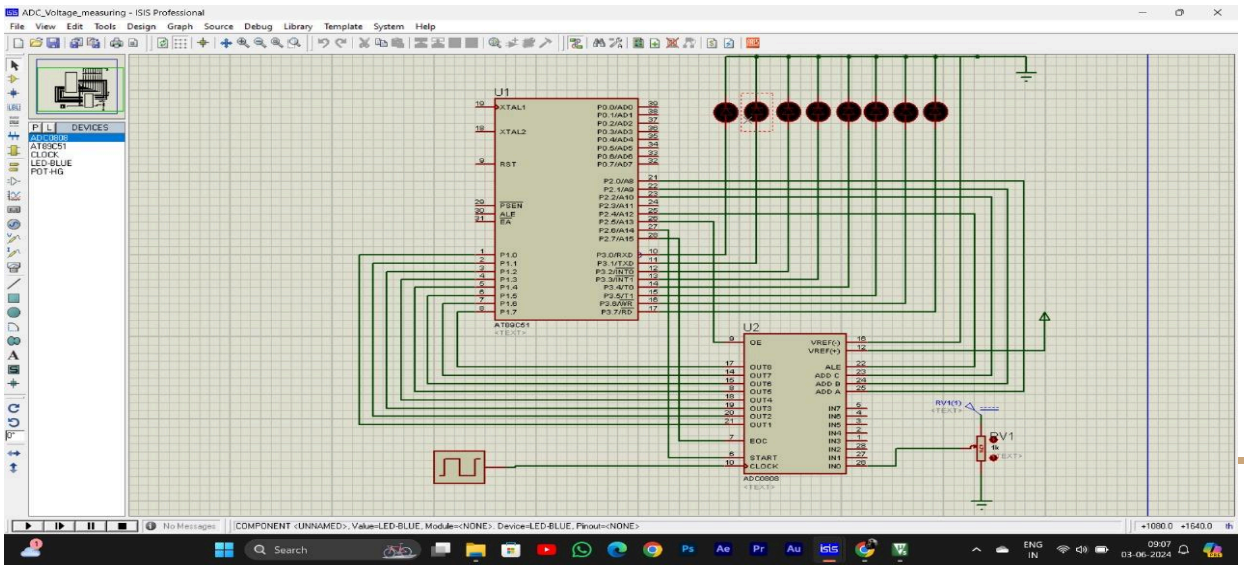
LEDs:

- Connect the anode of each LED to Port 3 (P3.0 to P3.7) of the 8051 through current-limiting resistors.
- Connect the cathode of each LED to GND.

CODE :

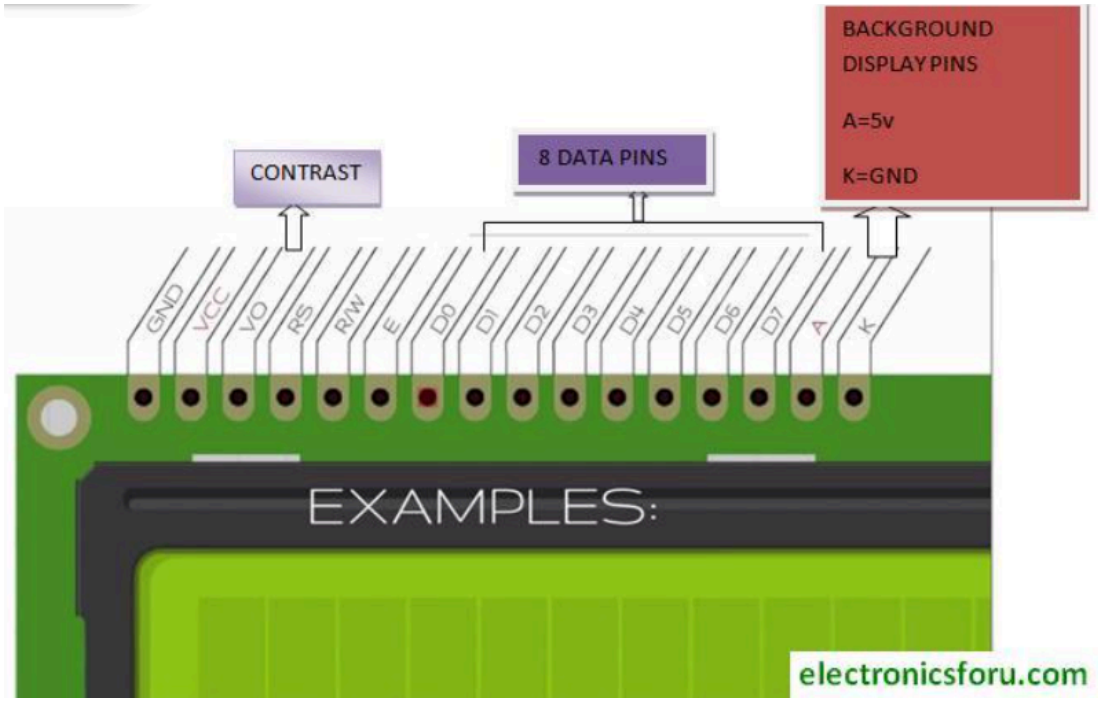
```
#include <reg51.h>
 sbit ALE = P2^4;
 sbit OE = P2^5;
 sbit SC = P2^6;
 sbit EOC = P2^7;
//Declaring the input selection pin
 sbit ADDR_A = P2^0;
 sbit ADDR_B = P2^1;
 sbit ADDR_C = P2^2;
//void MSDelay(unsigned int);
void MSDelay(unsigned int delay)
{
  unsigned int i,j;
  for(i=0;i<delay;i++)
  for(j=0;j<1275;j++);
}
void main()
{
  unsigned char ADC_Value = 0;
  P1 = 0xFF;
  EOC = 1;
  ALE = 0;
  OE = 0;
  SC = 0;
 while(1)
 {
  ADDR_C = 0;
  ADDR_B = 0;
  ADDR_A = 0;
  MSDelay(10);
  ALE = 1;
  MSDelay(10);
  SC = 1;
  MSDelay(10);
```

PROTEUS SIMULATION :



LCD :

16X2 LCD Display Pinout Diagram



The LCD 16x2 has a total of 16 pins, which can be divided into two groups: power pins and data pins. The power pins include:

VSS (ground): This pin is connected to the ground of the circuit.

VCC (power supply): This pin is connected to the positive power supply of the circuit, usually 5V.

VEE (contrast adjustment): This pin is used to adjust the contrast of the display. By adjusting the voltage on this pin, you can change the darkness of the characters on the screen.

The data pins include:

RS (register select): This pin is used to select the register that the data or commands that will be sent. When the RS (Register Select) signal is in a low state, data is transmitted to the instruction register (IR). On the other hand, when the RS signal is set to a high state, the data is directed to the data register (DR).

R/W (read/write): This pin is used to select whether the data is being written to the LCD or read from it. When R/W is low, the data is being written, and when it is high, the data is being read.

E (enable): This pin is used to enable the data transfer between the microcontroller and the LCD. A high-to-low transition on this pin initiates the data transfer.

DB0-DB7 (data bus): These pins are used to send data and commands to the LCD. The data is sent in parallel, with 8 bits being sent at a time.

Important Command Codes for 16×2 LCD

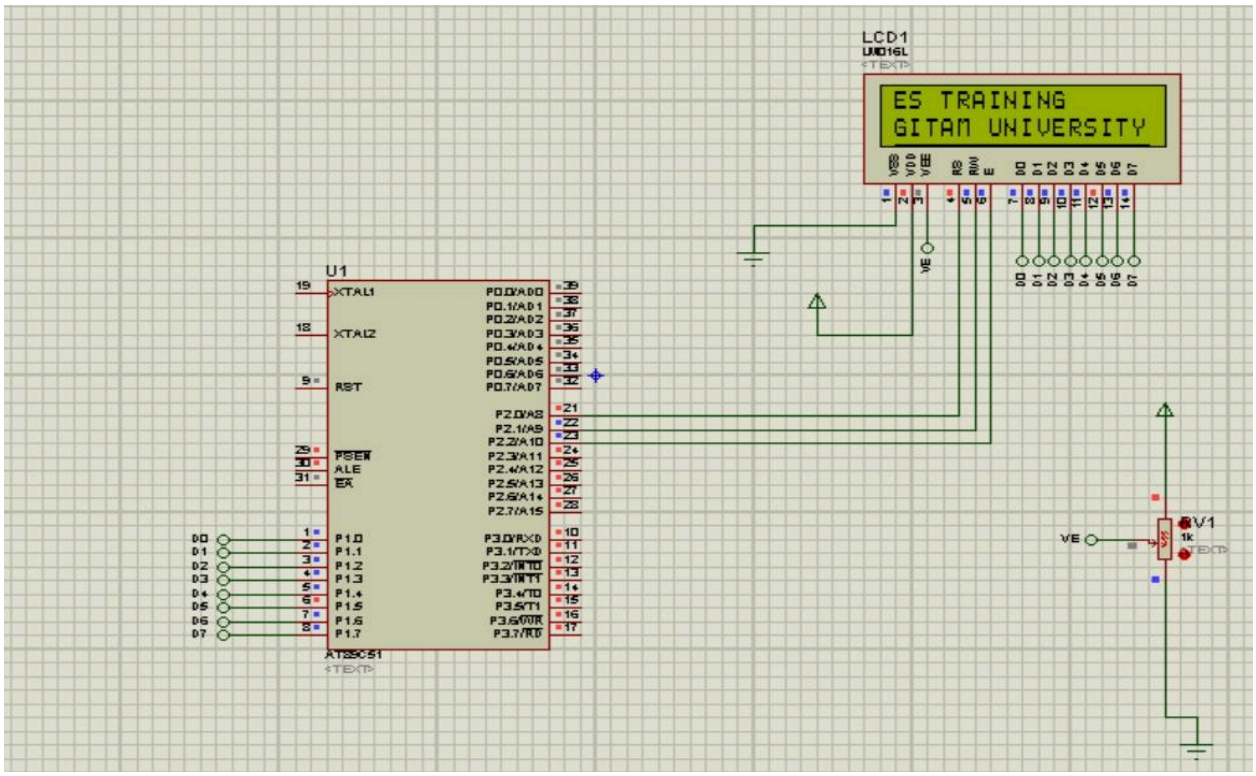
Sr.No.	Hex Code	Command to LCD instruction Register
1	01	Clear display screen
2	02	Return home
3	04	Decrement cursor (shift cursor to left)
4	06	Increment cursor (shift cursor to right)
5	05	Shift display right
6	07	Shift display left
7	08	Display off, cursor off
8	0A	Display off, cursor on
9	0C	Display on, cursor off
10	0E	Display on, cursor blinking
11	0F	Display on, cursor blinking
12	10	Shift cursor position to left
13	14	Shift the cursor position to the right
14	18	Shift the entire display to the left
15	1C	Shift the entire display to the right
16	80	Force cursor to the beginning (1st line)
17	C0	Force cursor to the beginning (2nd line)
18	38	2 lines and 5×7 matrix

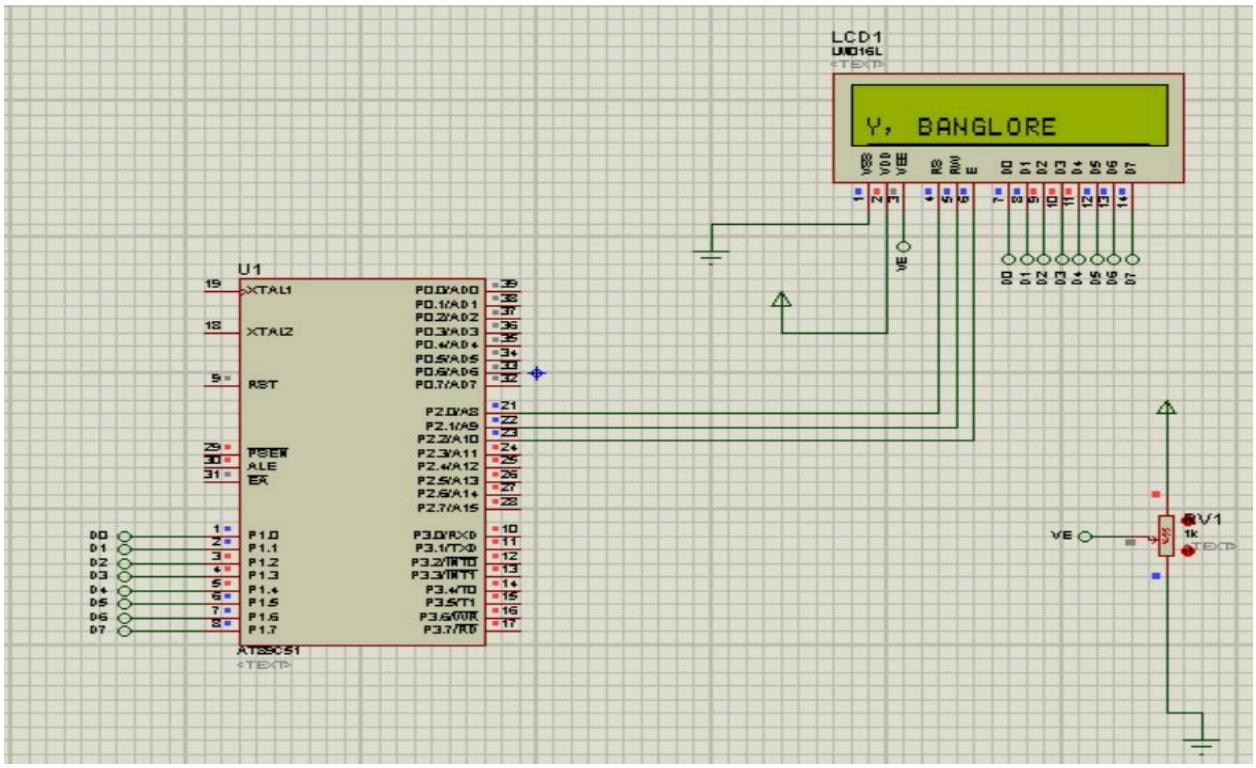
```
//Function for sending command
void WriteCommandToLCD(unsigned char ch)
{
   e=1;
   rs=0;
   rw=0;
   P1=ch;
   e=0;
   delay(20);
}
//Function for sending data
void WriteDataToLCD(unsigned char ch)
{
   e=1;
   rs=1;
   rw=1;
   P1=ch;
   e=0;
   delay(20);
}
```

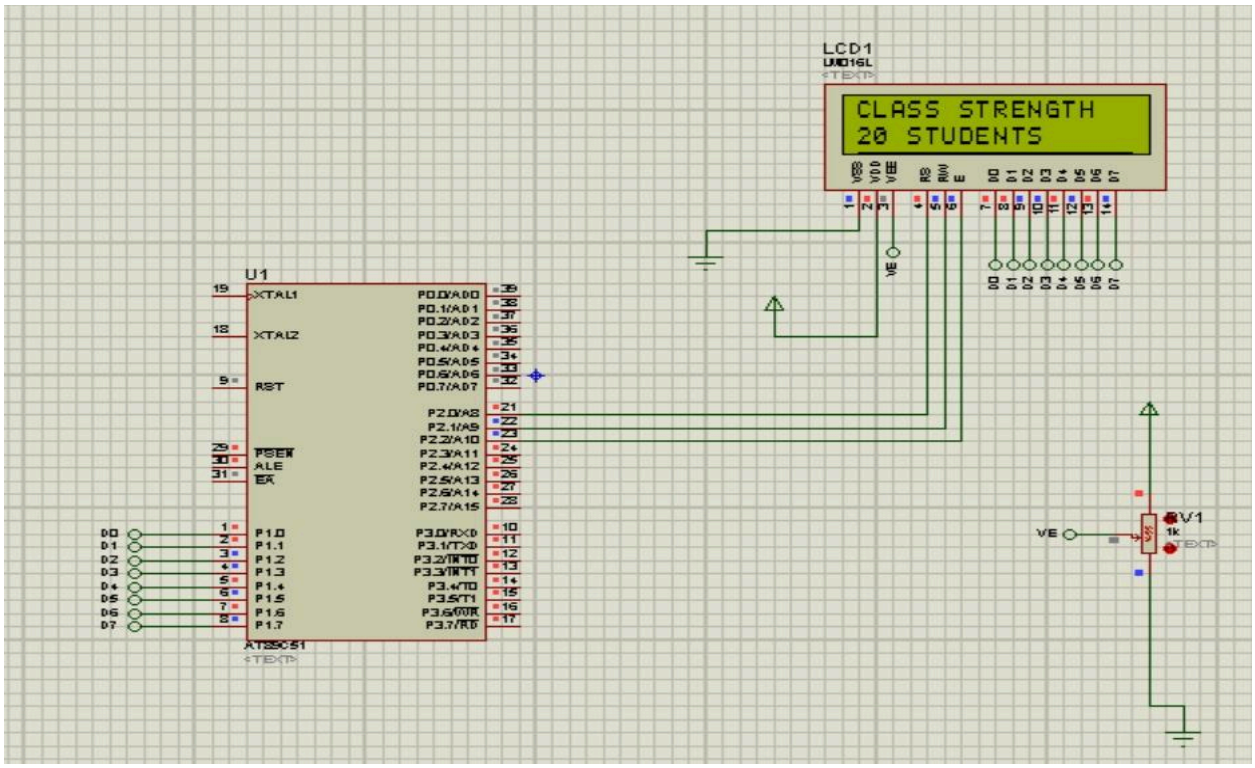
CODE :

```
#include<reg51.h>
sbit rs=P2^0;
sbit rw=P2^1;
sbit e=P2^2;
void delay(unsignesd int);
void WriteCommandToLCD(unsigned char ch);
void WriteDataToLCD(unsigned char ch);
void WriteStringToLCD(unsigned char ch[]);
void mai(void)
{
  unsigned char ch1[]="GITAM UNIVERSITY, BANGLORE";
  unsigned char j,k;
  unsigned int MyData = 20;

//LCD Initialization
  WriteCommandToLCD(0x38);
  WriteCommandToLCD(0x01);
  WriteCommandToLCD(0x0E);
  WriteCommandToLCD(0x80);
  WriteCommandToLCD(0x06);

//Sending Data to LCD
  WriteStringToLCD("ES Training");
  WriteCommandToLCD(0xC0);
  for(j=0;ch!='\0';j++)
  {
    WriteDataToLCD(ch1[j]);
  }
  for(k=0;k<30;k++)
  {
    WriteCommandToLCD(0x1c);
  }
  while(1)
    {
      WriteCommandToLCD(0x01);
      WriteCommandToLCD(0x08);
      WriteStringToLCD("CLASS STRENGTH");
      WriteCommandToLCD(0xc0);
      WriteDataToLCD((MyData / 10)+ 48);
      WriteDataToLCD((MyData % 10)+ 48);
      WriteStringToLCD("STUDENTS");
    }
    //}
    }
void delay(unsigned int t)
  {
    unsigned int i,j;
    for(i=0;i<t;i++)
    for(j=0;j<1275;j++);
  }
   void WriteCommandToLCD(unsigned char ch)
     {
       e=1;
       rs=0;
       rw=0;
       P1=ch;
       e=0;
       delay(20);
     }
      void WriteDataToLCD(unsigned char ch)
        {
          e=1;
          rs=1;
          rw=0;
          P1=ch;
          e=0;
          delay(20);
        }
        void WriteStringToLCD(unsigned char ch[])
          {
            int i;
            for(i=0;ch[i]!='\0';i++)
            {
              WriteDataToLCD(ch[i]);
            }
          }
```

PROTEUS SIMULATION:







K.Gouthami

[BU21EECE0100498]