Department of School of Technology

Lab Manual

SUBJECT: Embedded Systems (18IC311T)

6th Semester (B. Tech)

(Branch: ICT)

Prepared By: Team 5

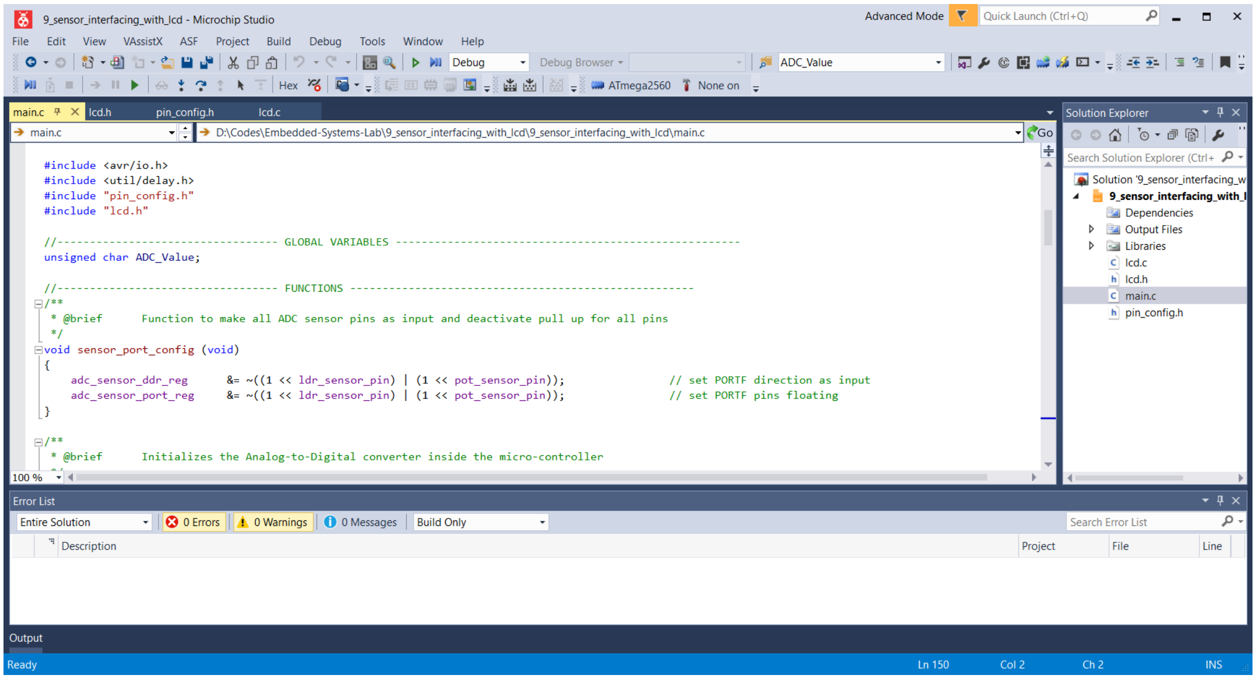
Jay Bhgaiya (18BIT040)
Jehil Thakkar (18BIT043)
Jai Lohana (18BIT035)
Deep Mistry (18BIT015)
Kathan Pathak (18BIT051)

Pandit Deendayal Energy University
School of Technology, Gandhinagar - 382426, Gujarat

List of Experiments

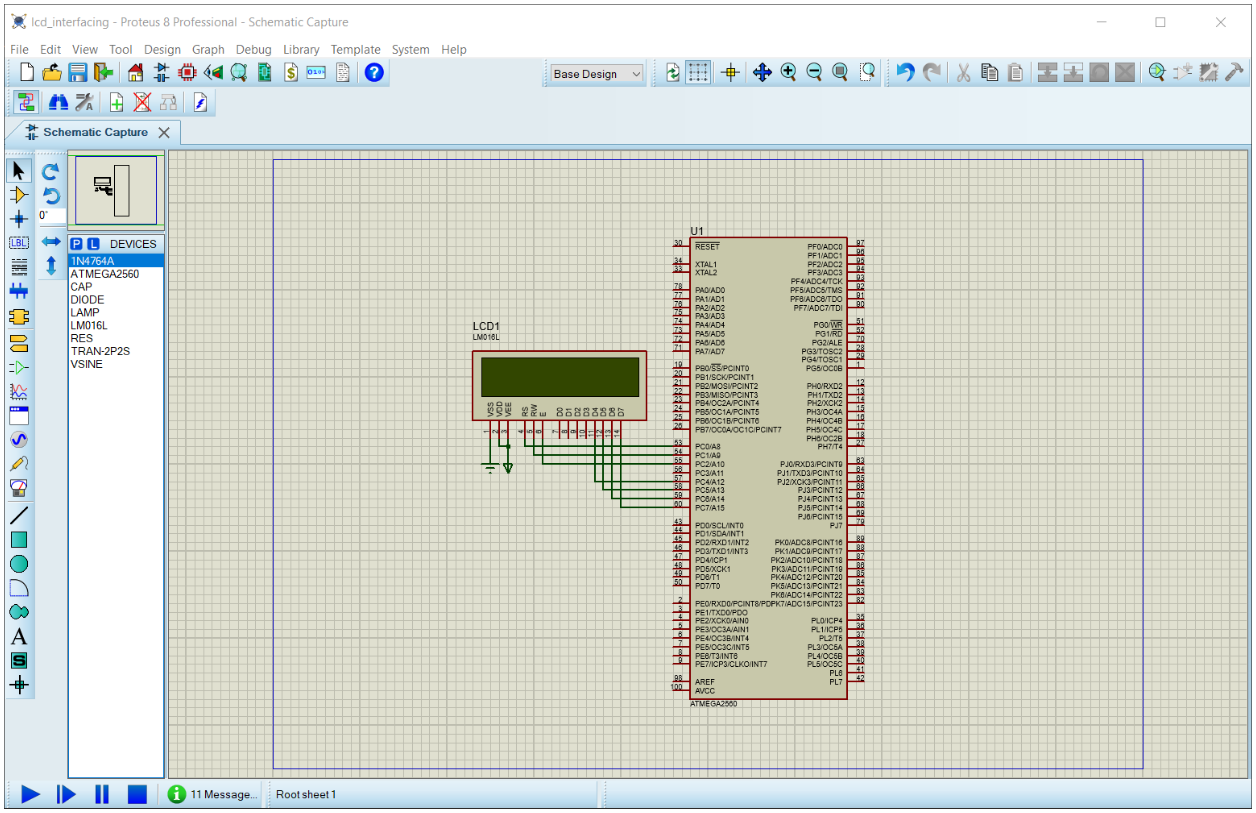
1. Familiarization with IDE and trainer kits/boards.

Atmel Studio: For Programming

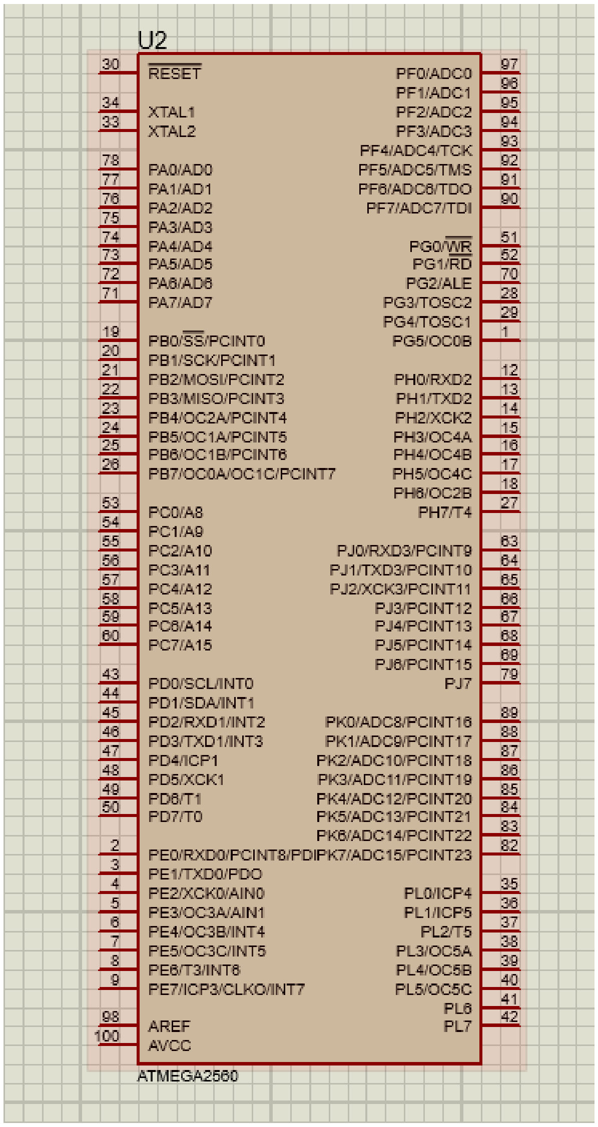


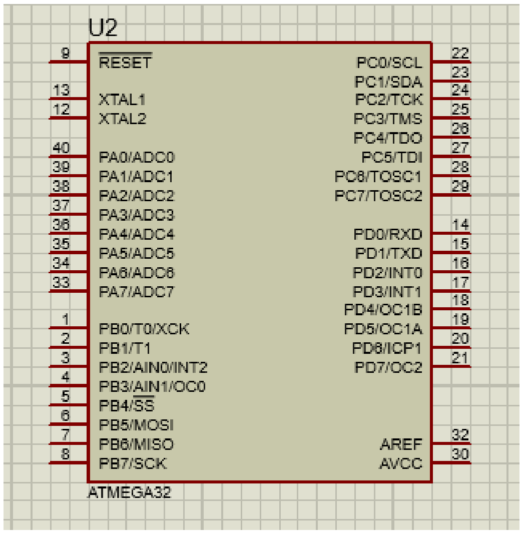
```c
#include <avr/io.h>
#include <util/delay.h>
#include "pin_config.h"
#include "lcd.h"

//--------------------------------- GLOBAL VARIABLES ---------------------------------------------------
unsigned char ADC_Value;

//--------------------------------- FUNCTIONS ---------------------------------------------------
/**
 * @brief      Function to make all ADC sensor pins as input and deactivate pull up for all pins
 */
void sensor_port_config (void)
{
    adc_sensor_ddr_reg      &= ~((1 << ldr_sensor_pin) | (1 << pot_sensor_pin));              // set PORTF direction as input
    adc_sensor_port_reg     &= ~((1 << ldr_sensor_pin) | (1 << pot_sensor_pin));              // set PORTF pins floating
}

/**
 * @brief      Initializes the Analog-to-Digital converter inside the micro-controller
```

Proteus 8: For Simulation



AVR Micro-Controller: atmega2560 and atmega32



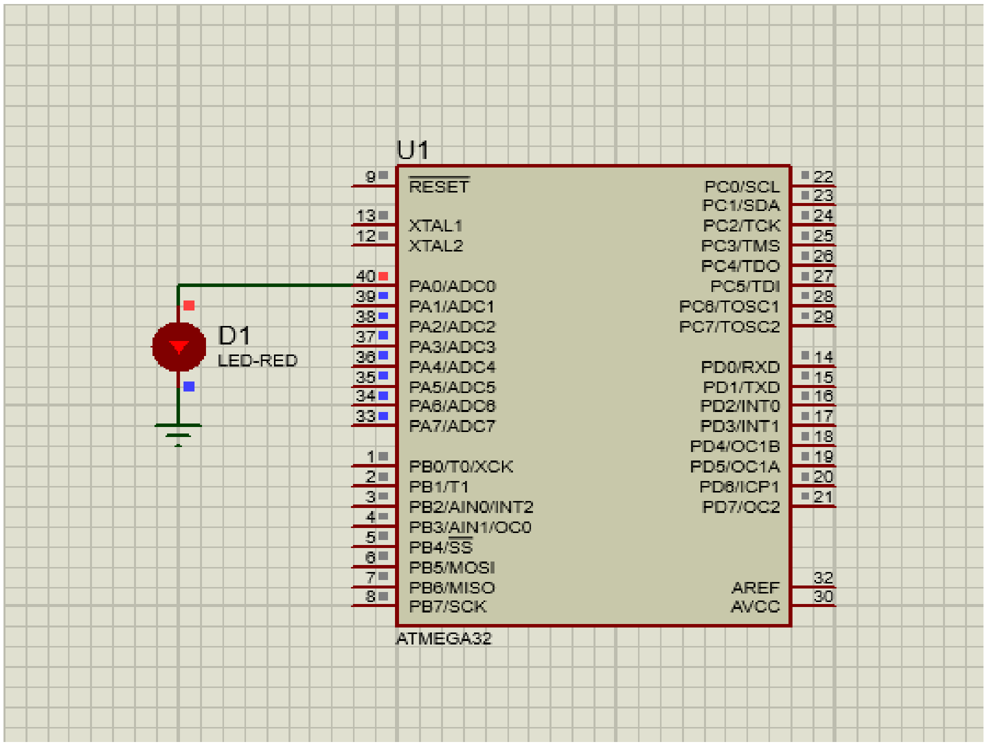


2. Program for blinking LED, pattern generation, timing, sequence generation.

a) LED Blinking:

```
#ifndef F_CPU
#define F_CPU 16000000UL //clock speed is 16MHz
#endif

#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRA = 0xFF;
    while (1)
    {
              PORTA = 0x01;
              _delay_ms(1000);
              PORTA = 0x00;
              _delay_ms(1000);
    }
}
```

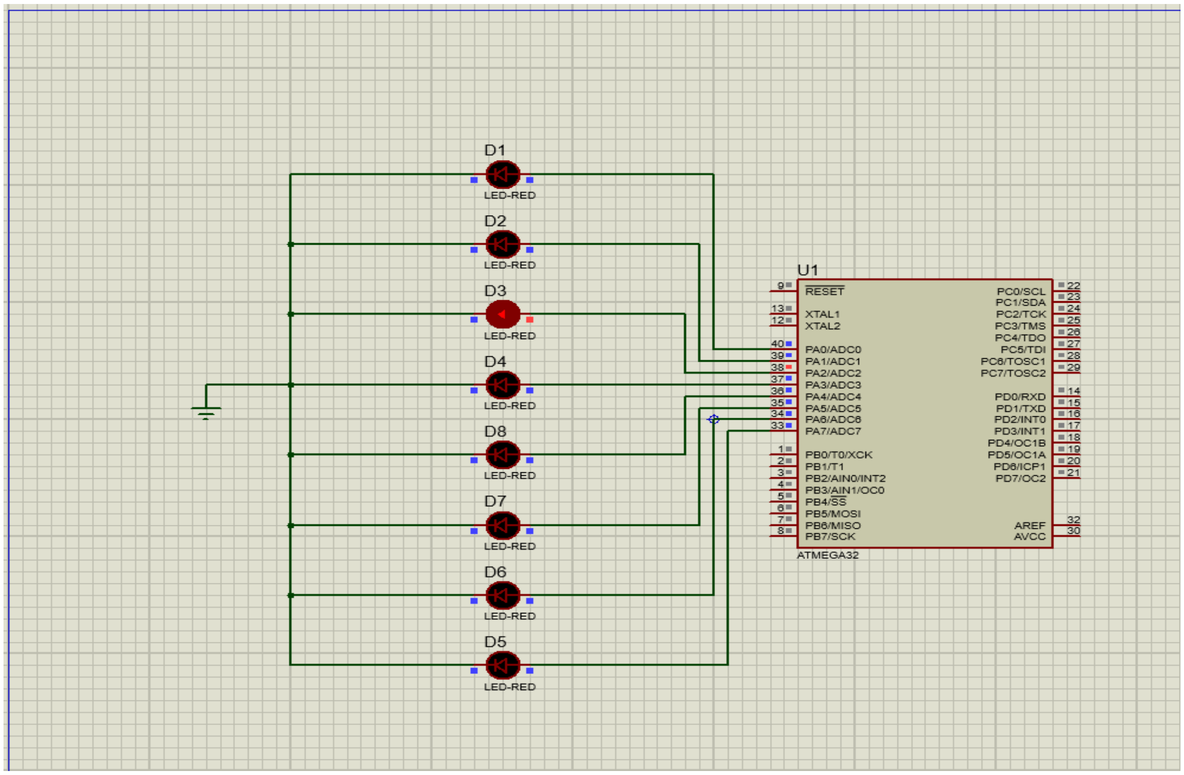
Output:



b) LED Blinking 1 to 8 (Pattern 1)

```
#ifndef F_CPU
#define F_CPU 16000000UL //clock speed is 16MHz
#endif

#include <avr/io.h>
#include <util/delay.h>


int main(void)
{
    DDRA = 0xFF;
    while (1)
    {
              for (int i=0; i<8; i++)
              {
                     PORTA = (0x01 << i);
                    _delay_ms(500);
                     PORTA = 0x00;
              }
    }
}
```

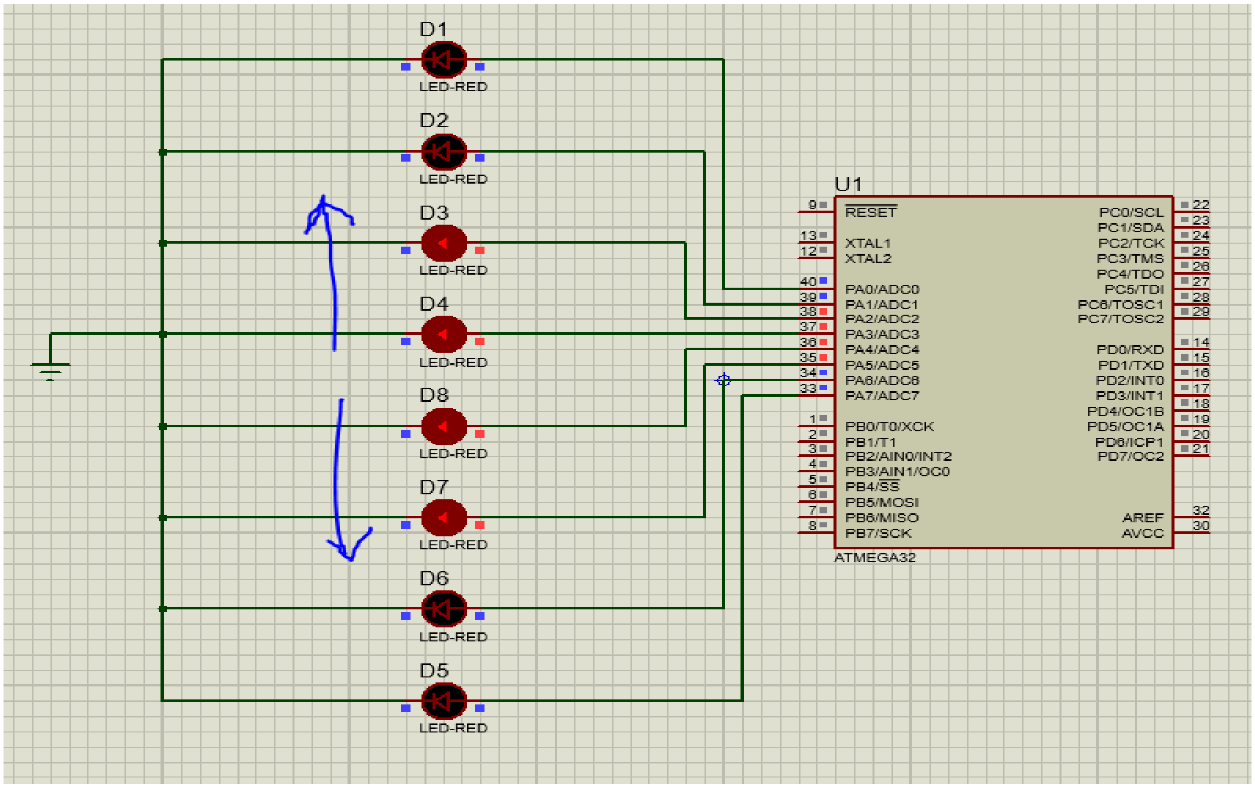
Output:



c) Led Blinking From Mid (Pattern 2)

```
#ifndef F_CPU
#define F_CPU 16000000UL //clock speed is 16MHz
#endif
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRA = 0xFF;
    while (1)
    {
              for (int i=3; i>=0; i--)
              {
                     PORTA |= (1 << i);
                     PORTA |= (1 << (7-i));
                     _delay_ms(500);
              }
              for (int i=0; i<4; i++)
              {
                     PORTA &= ~(1 << i);
                     PORTA &= ~(1 << (7-i));
                     _delay_ms(500);
              }
    }
}
```

Output:

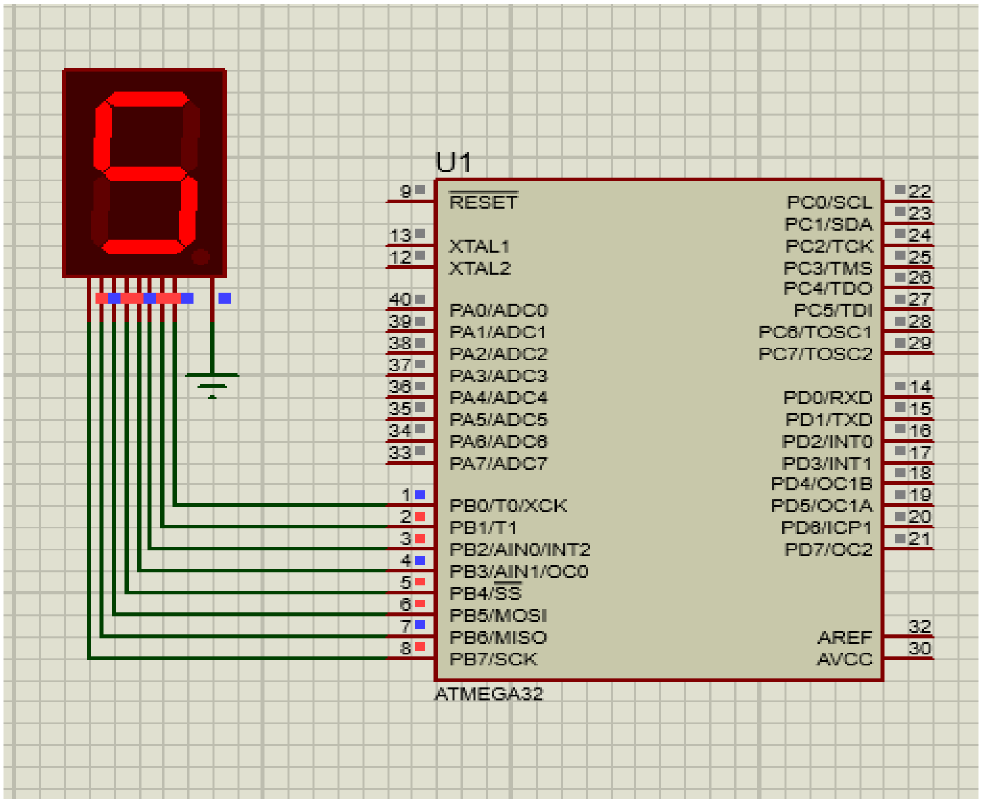


3. Program for interfacing multi-digit 7 segment display and implementing counter.

a) Seven Segment LED

```
#ifndef F_CPU
#define F_CPU 16000000UL //clock speed is 16MHz
#endif
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRB = 0xFF;
    while (1)
    {
              char array[] = {0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE0, 0xFE,
0xF6};
              for (int i=0; i<=9; i++)
              {
                     PORTB = array[i];
                     _delay_ms(1000);
              }
    }
}
```

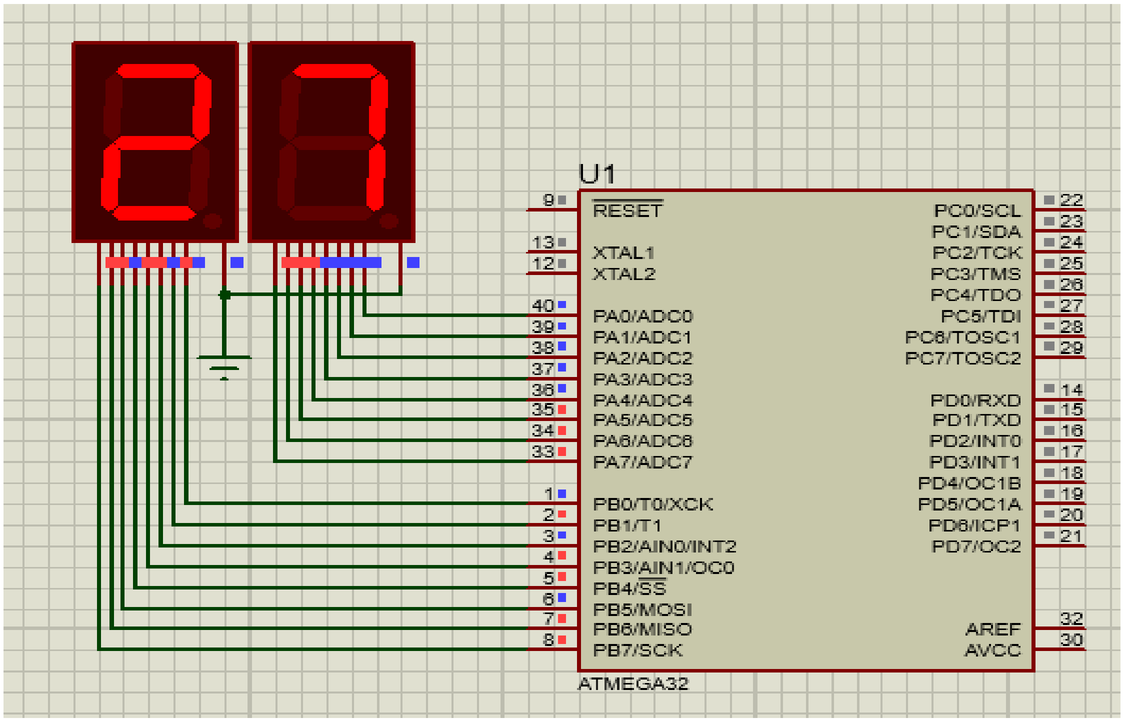
Output:



b) Seven Segment LED (Counter)

```
#ifndef F_CPU
#define F_CPU 16000000UL //clock speed is 16MHz
#endif
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRB = 0xFF;
       DDRA = 0xFF;

       char array[] = {0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE0, 0xFE, 0xF6};
       int tenth_num = 0;
       while (1)
    {
              PORTB = array[tenth_num++];

              for(int i=0; i<10; i++)
              {
                     PORTA = array[i];
                     _delay_ms(500);
              }

              if(tenth_num == 10)
                     tenth_num = 0;
    }
}
```

Output:

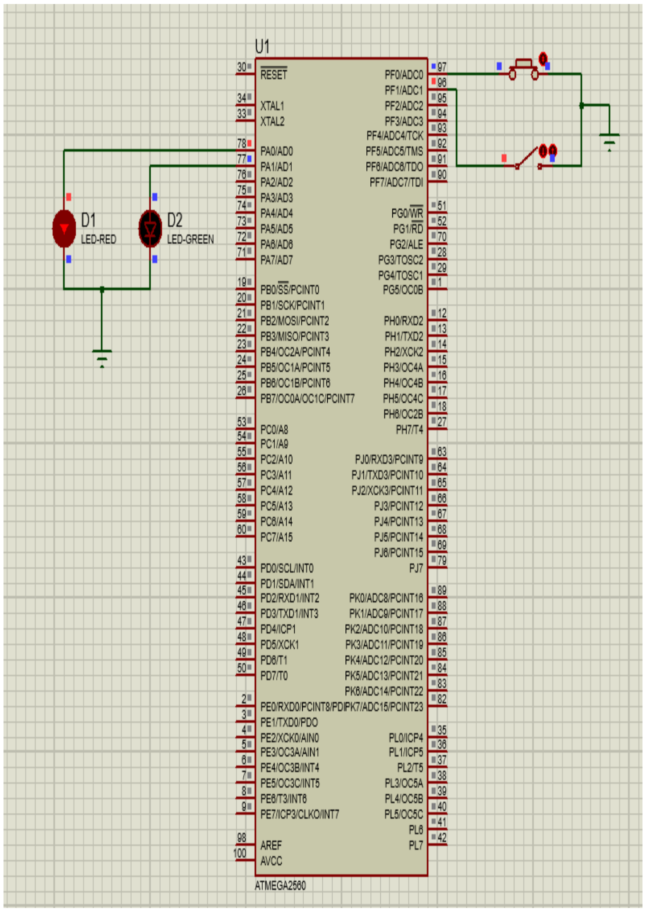


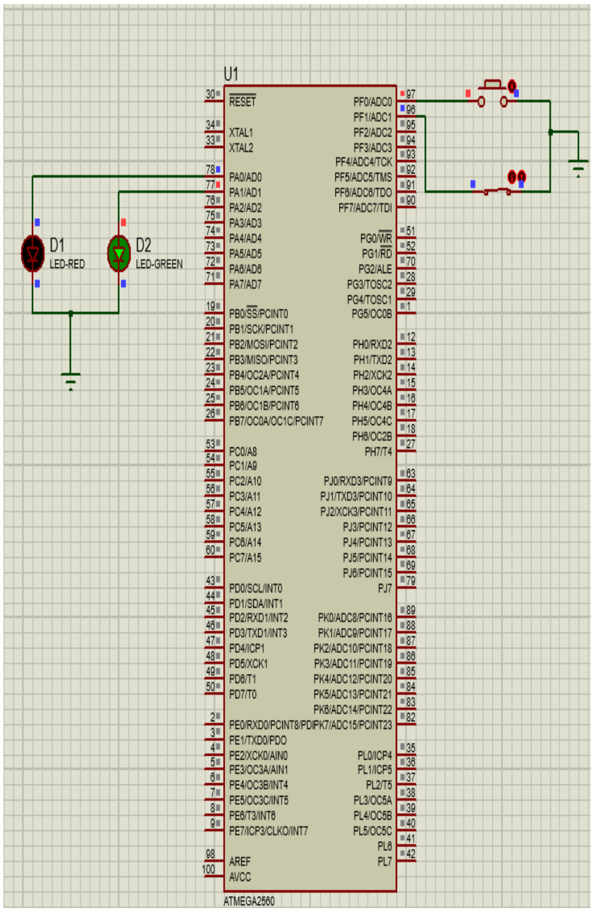
4. Program for interfacing toggle and push button switches.

```
#include <avr/io.h>
#include <util/delay.h>

// Definitions for ATmega2560 and Interfacing of Toggle and push button
#if defined(__AVR_ATmega2560__)

       #define toggle_button_ddr_reg            DDRF
       #define toggle_button_port_reg           PORTF
       #define toggle_button_pin_reg            PINF
       #define toggle_button_pin                PF1

       #define push_button_ddr_reg              DDRF
       #define push_button_port_reg             PORTF
       #define push_button_pin_reg              PINF
       #define push_button_pin                         PF0

       #define led_ddr_reg                             DDRA
       #define led_port_reg                     PORTA
       #define led_1_pin                               PA0
       #define led_2_pin                               PA1

#endif

//---------------------------------- FUNCTIONS ------------------------------------------
---------------

//-----------------------------CONFIGURATION FUNCTIONS ----------------------------------
----------------

/**
 * @brief      Function to make **ONLY** Toggle and push button Switch pin as input and
pull it up internally
 */
void toggle_and_push_button_switch_config (void) {

       // Make **ONLY** Toogle Switch pin as input
       toggle_button_ddr_reg &= ~( 1 << toggle_button_pin );

       // Make **ONLY** Toggle Switch pin internally pull-up
       toggle_button_port_reg |= ( 1 << toggle_button_pin );

       // Make **ONLY** Push button Switch pin as input
       push_button_ddr_reg &= ~( 1 << push_button_pin );

       // Make **ONLY** Push button Switch pin internally pull-up
       push_button_port_reg |= ( 1 << push_button_pin );

}

/**
 * @brief      Function to make **ONLY** 'led_1_pin' and 'led_2_pin' as output and
initially set it to low
 */
void led_pin_config (void) {
       // Make 'led_1_pin' as output
       led_ddr_reg   |= ( 1 << led_1_pin );

       // Set 'led_1_pin' to low initially
       led_port_reg &= ~( 1 << led_1_pin );

       // Make 'led_2_pin' as output
       led_ddr_reg   |= ( 1 << led_2_pin );

       // Set 'led_2_pin' to low initially
       led_port_reg &= ~( 1 << led_2_pin );
}

//------------------------------------- LED RELATED FUNCTIONS --------------------------
--------------

/**
 * @brief      Function to set LED 1 pin to high, hence turn on LED 1
 */
void led_1_on(void){
       // Turn on all LEDs
       led_port_reg |= (1 << led_1_pin);
}

/**
 * @brief      Function to set LED 1 pin to low, hence turn off LED 1
 */
void led_1_off(void){
       // Turn off all LEDs
       led_port_reg &= ~(1 << led_1_pin);
}

/**
 * @brief      Function to set LED 2 pin to high, hence turn on LED 2
 */
void led_2_on(void){
       // Turn on all LEDs
       led_port_reg |= (1 << led_2_pin);
}

/**
 * @brief      Function to set LED 2 pin to low, hence turn off LED 2
 */
void led_2_off(void){
       // Turn off all LEDs
       led_port_reg &= ~(1 << led_2_pin);
}

//---------------------------------- MAIN ----------------------------------------------
-----------------

int main(void)
{
       // Initialize the necessary devices (Led, Toggle and push button switch) required
for the experiment.
       toggle_and_push_button_switch_config();
       led_pin_config();
```

```
    while (1)
    {
              // If the push button Switch is NOT pressed
              if ((push_button_pin_reg & (1 << push_button_pin)) == (1 <<
push_button_pin)) {
                     _delay_ms(100);
                     led_1_off();                 //Turn off LED 1
              }
              else {
                     _delay_ms(100);
                     led_1_on();                  //Turn on LED 1
              }
              if ((toggle_button_pin_reg & (1 << toggle_button_pin)) == (1 <<
toggle_button_pin)) {
                     _delay_ms(100);
                     led_2_off();                 //Turn off LED 2
              }
              else {
                     _delay_ms(100);
                     led_2_on();                  //Turn on LED 2
              }
    }
}
```

Output:

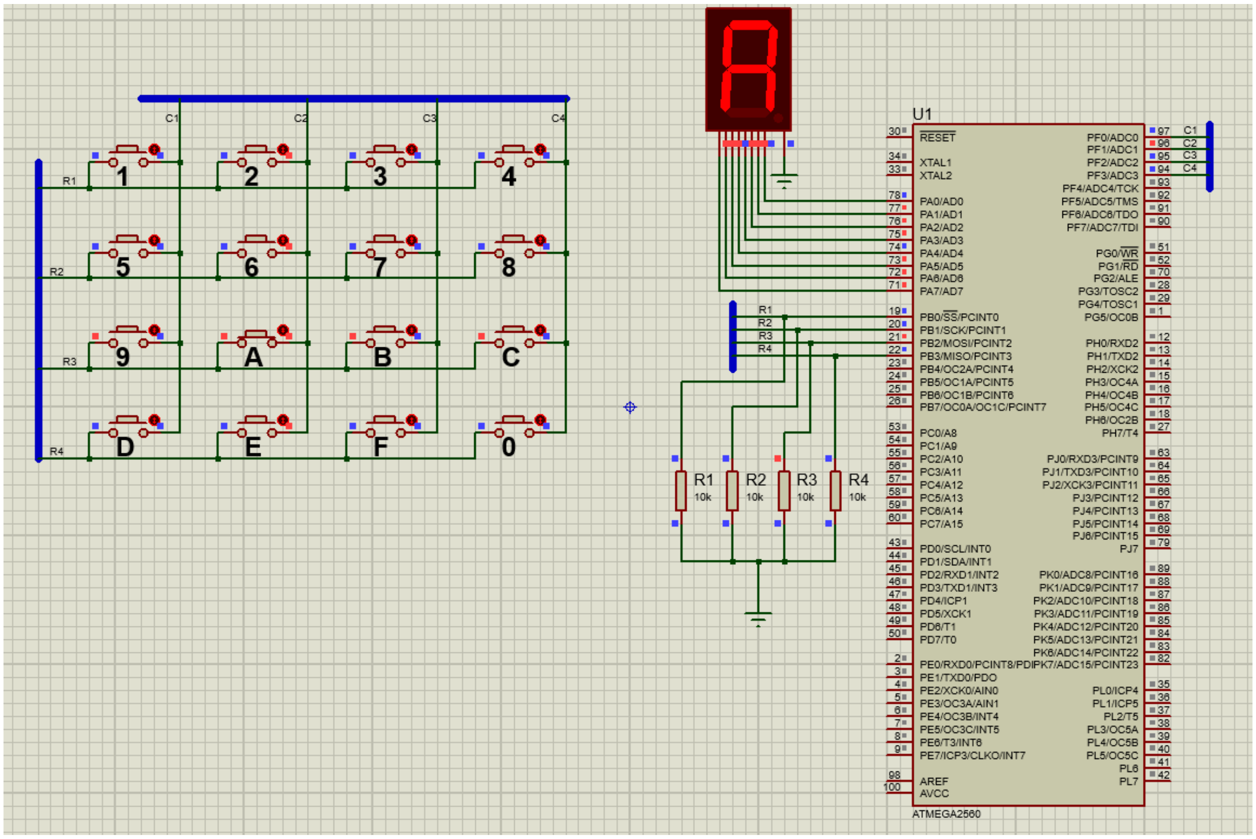




5. Program for interfacing simple keypad and matrix keypad, controlling LEDs using switches.

```
#include <avr/io.h>
#include <util/delay.h>

#if defined(__AVR_ATmega2560__)

       #ifndef F_CPU
       #define F_CPU 14745600
       #endif

       //---------------------------------- INPUT / OUTPUT PERIPHERALS ------------------
-----------------------------------

       // Seven Segment definitions
       #define seven_seg_ddr_reg          DDRA
       #define seven_seg_port_reg         PORTA

       // Matrix Column definitions
       #define matrix_col_ddr_reg         DDRF
       #define matrix_col_port_reg        PORTF
       #define matrix_col_1_pin           PF0
       #define matrix_col_2_pin           PF1
       #define matrix_col_3_pin           PF2
       #define matrix_col_4_pin           PF3

       // Matrix Row Definitions
       #define matrix_row_ddr_reg         DDRB
       #define matrix_row_port_reg        PORTB
       #define matrix_row_pin_reg         PINB
       #define matrix_row_1_pin           PB0
       #define matrix_row_2_pin           PB1
       #define matrix_row_3_pin           PB2
       #define matrix_row_4_pin           PB3

#endif
// ----------------------------------------- Functions ----------------------------------
-------

//-----------------------------CONFIGURATION FUNCTIONS ----------------------------------
----------------

/**
 * @brief      Function to make **ONLY** Configuration of Seven Segment Display.
 */
void seven_seg_config(void)
{
       // Make **ONLY** all pins as output
       seven_seg_ddr_reg = 0xFF;

       // Make **ONLY** set all pins initially low
       seven_seg_port_reg = 0x00;
}

/**
 * @brief      Function to make **ONLY** Configuration of Columns of matrix.
 */
void matrix_col_pin_config(void)
{
       // Make **ONLY** Four column pins as output
       matrix_col_ddr_reg |= ((1 << matrix_col_1_pin) | (1 << matrix_col_2_pin) | (1 <<
matrix_col_3_pin) | (1 << matrix_col_4_pin));

       // Make **ONLY** Disabling all columns witout disturbing reamianing pins
       matrix_col_port_reg &= ~((1 << matrix_col_1_pin) | (1 << matrix_col_2_pin) | (1 <<
matrix_col_3_pin) | (1 << matrix_col_4_pin));
}

/**
 * @brief      Function to make **ONLY** Configuration of Rows of matrix.
 */
void matrix_row_pin_config(void)
{
       // Make **ONLY** Four row pins defined as input
       matrix_row_ddr_reg &= ~((1 << matrix_row_1_pin) | (1 << matrix_row_2_pin) | (1 <<
matrix_row_3_pin) | (1 << matrix_row_4_pin));

       // Make **ONLY** Disabling all pull-up resister on four rows witout disturbing
reamianing pins
       matrix_row_port_reg &= ~((1 << matrix_row_1_pin) | (1 << matrix_row_2_pin) | (1 <<
matrix_row_3_pin) | (1 << matrix_row_4_pin));
}

// ----------------------------------------- Main ---------------------------------------
--
int main(void)
{
       seven_seg_config();
       matrix_col_pin_config();
       matrix_row_pin_config();

     while (1)
     {
            char array[] = {0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE0, 0xFE, 0xF6,
0xEE, 0x3E, 0x9C, 0x7A, 0x9E, 0x8E};
                         //   {   0,    1,    2,    3,    4,    5,    6,    7,    8,
9,    A,    B,    C,    D,    E,    F}
               int key=0, column = 0, temp = 0;
            for (column=1,temp=1; column<=4; temp*=2,column++)
            {
                      // Make **ONLY** Disabling all pull-up resister on four rows witout
disturbing reamianing pins
                      matrix_col_port_reg &= ~((1 << matrix_col_1_pin) | (1 <<
matrix_col_2_pin) | (1 << matrix_col_3_pin) | (1 << matrix_col_4_pin));
                      matrix_col_port_reg |= temp;

                      // Reading rows data and identify the key
                      switch(matrix_row_pin_reg & ((1 << matrix_row_1_pin) | (1 <<
matrix_row_2_pin) | (1 << matrix_row_3_pin) | (1 << matrix_row_4_pin)))
                      {
                             case (1 << matrix_row_1_pin):{                 // row1
                                    key = column;
                             }break;
                             case (1 << matrix_row_2_pin):{                  // row2
                                    key = 4 + column;
                             }break;

                             case (1 << matrix_row_3_pin):{                  // row3
                                    key = 8 + column;
                             }break;

                             case (1 << matrix_row_4_pin):{                  // row4
                                    key = 12 + column;
                             }break;
                      }
                  _delay_ms(10);  // Key debounce
            }
               if((key < 0) || (key > 15) ){
                      seven_seg_port_reg = array[0];
               }
               else{
                      seven_seg_port_reg = array[key];
               }
     }
}
```

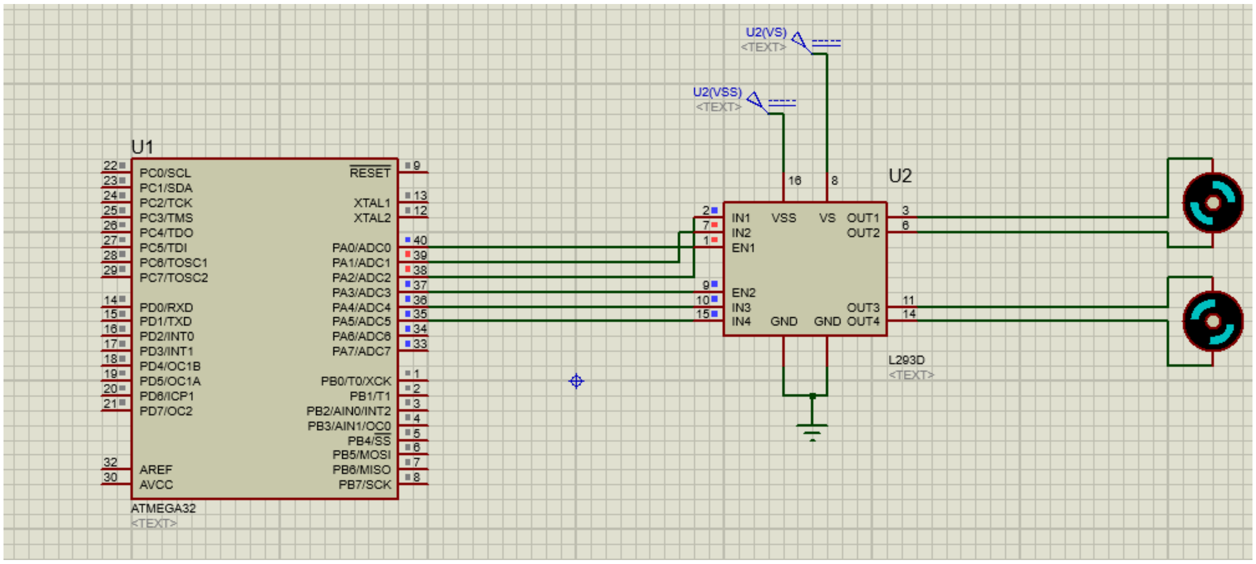
Output:



6. Program for interfacing DC motor with AVR microcontroller.

```
#include<stdio.h>
#define F_CPU 16000000UL
#include<avr/io.h>
#include<util/delay.h>


int main(void){
        DDRA = 0xFF;
        while(1)
        {
                PORTA =0x00;
                _delay_ms(100);
                PORTA =0x06;
                _delay_ms(100);
                PORTA =0x28;
                _delay_ms(100);
                PORTA =0x1E;
                _delay_ms(100);
                PORTA =0x2D;
                _delay_ms(100);
        }
}
```

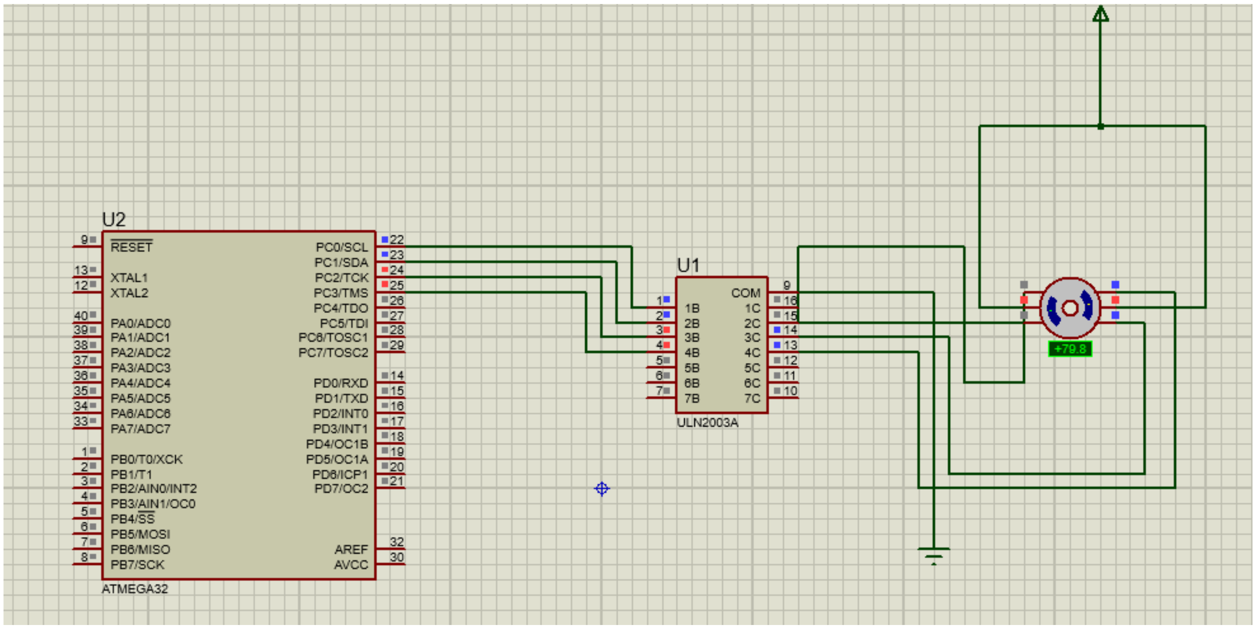
Output:



7. Program for interfacing stepper motor with AVR microcontroller.

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{

       int period;
       DDRC = 0x0F;         /* Make PORTD lower pins as output */
       period = 100;        /* Set period in between two steps */
       while (1)
       {
              /* Rotate Stepper Motor clockwise with Half step sequence */
              for(int i=0;i<12;i++)
              {
                     PORTC = 0x09;
                     _delay_ms(period);
                     PORTC = 0x08;
                     _delay_ms(period);
                     PORTC = 0x0C;
                     _delay_ms(period);
                     PORTC = 0x04;
                     _delay_ms(period);
                     PORTC = 0x06;
                     _delay_ms(period);
                     PORTC = 0x02;
                     _delay_ms(period);
                     PORTC = 0x03;
                     _delay_ms(period);
                     PORTC = 0x01;
                     _delay_ms(period);
              }
              PORTC = 0x09;        /* Last step to initial position */
              _delay_ms(period);
              _delay_ms(1000);

              /* Rotate Stepper Motor Anticlockwise with Full step sequence */
              for(int i=0;i<12;i++)
              {
                     PORTC = 0x09;
                     _delay_ms(period);
                     PORTC = 0x03;
                     _delay_ms(period);
                     PORTC = 0x06;
                     _delay_ms(period);
                     PORTC = 0x0C;
                     _delay_ms(period);
              }
              PORTC = 0x09;
              _delay_ms(period);
              _delay_ms(1000);
       }
}
```

Output:

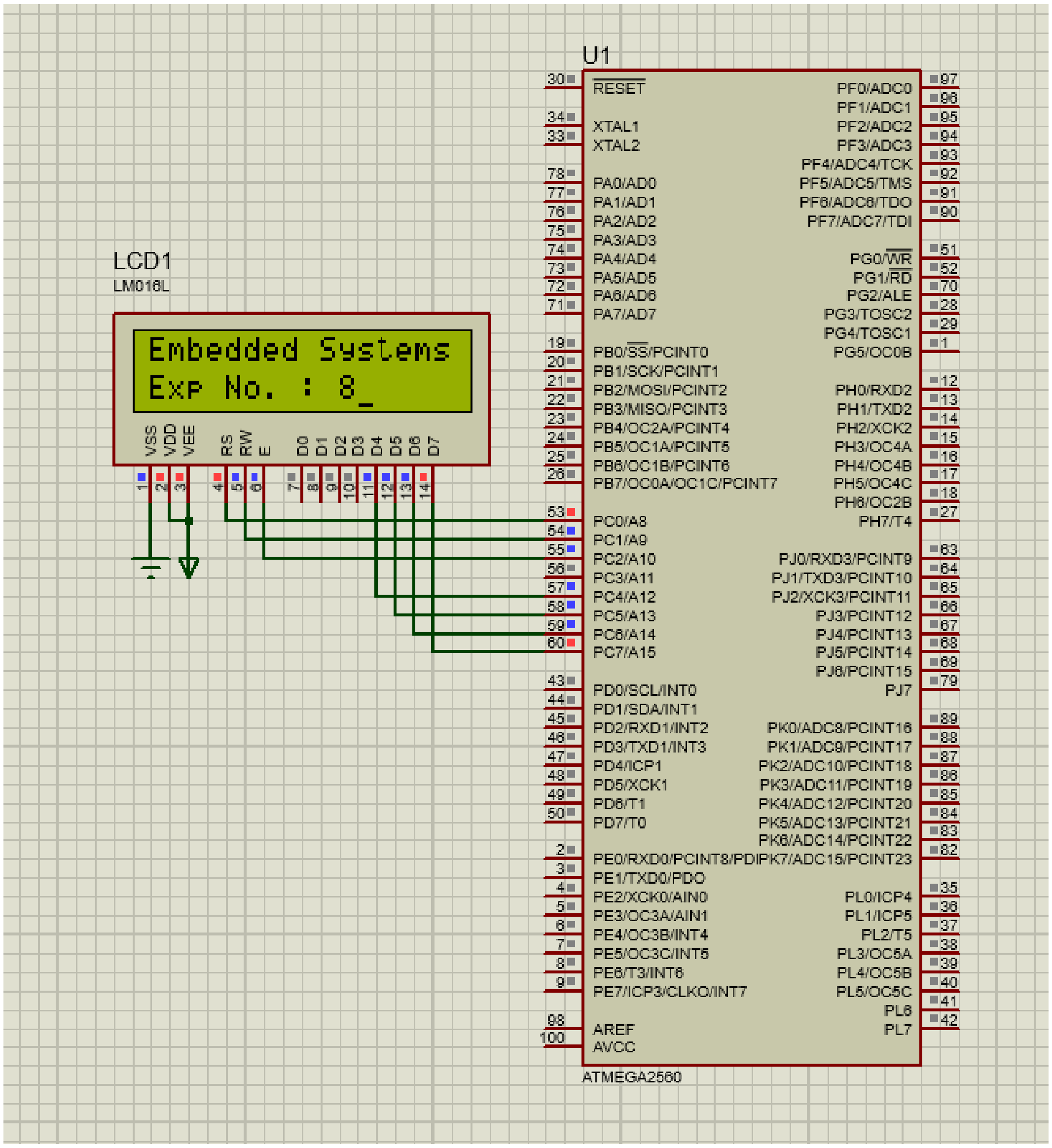


8. Program for interfacing LCD and displaying text on it.

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <stdio.h>
#include <util/delay.h>

void command (unsigned char cmd)
{
	PORTC = 0X02;
	PORTD = cmd;
	PORTC = 0X00;
	_delay_ms(15);
}
void lcd_data(unsigned char data)
{
	PORTC = 0X03;
	PORTD = data;
	PORTC = 0X01;
	_delay_ms(15);
}
void lcd_print(char *p)
{
	while(*p)
	{
		lcd_data(*p++);
	}
}
int main(void)
{
	DDRC=0XFF;//This register is used for selecting the R/S and R/W pin.
```

```
    DDRD=0XFF;//This register is used to give the data or commands.
    command(0x38);//Activated 2 lines in 8-bit mode.
    command(0X0F);//Display is ON, cursor is blinking.
    command(0x01);//Clearing the display.
    while(1)
    {
        command(0X80);//Forced the cursor to first position of first line.
        lcd_print("Embedded Systems");
        _delay_ms(1000);
        command(0XC0);//Forced the cursor to the first position of second line.
        lcd_print("Exp No. : 8");
        _delay_ms(1000);
        command(0X01);//Clearing the display.
        _delay_ms(1000);
    }
}
```

Output:



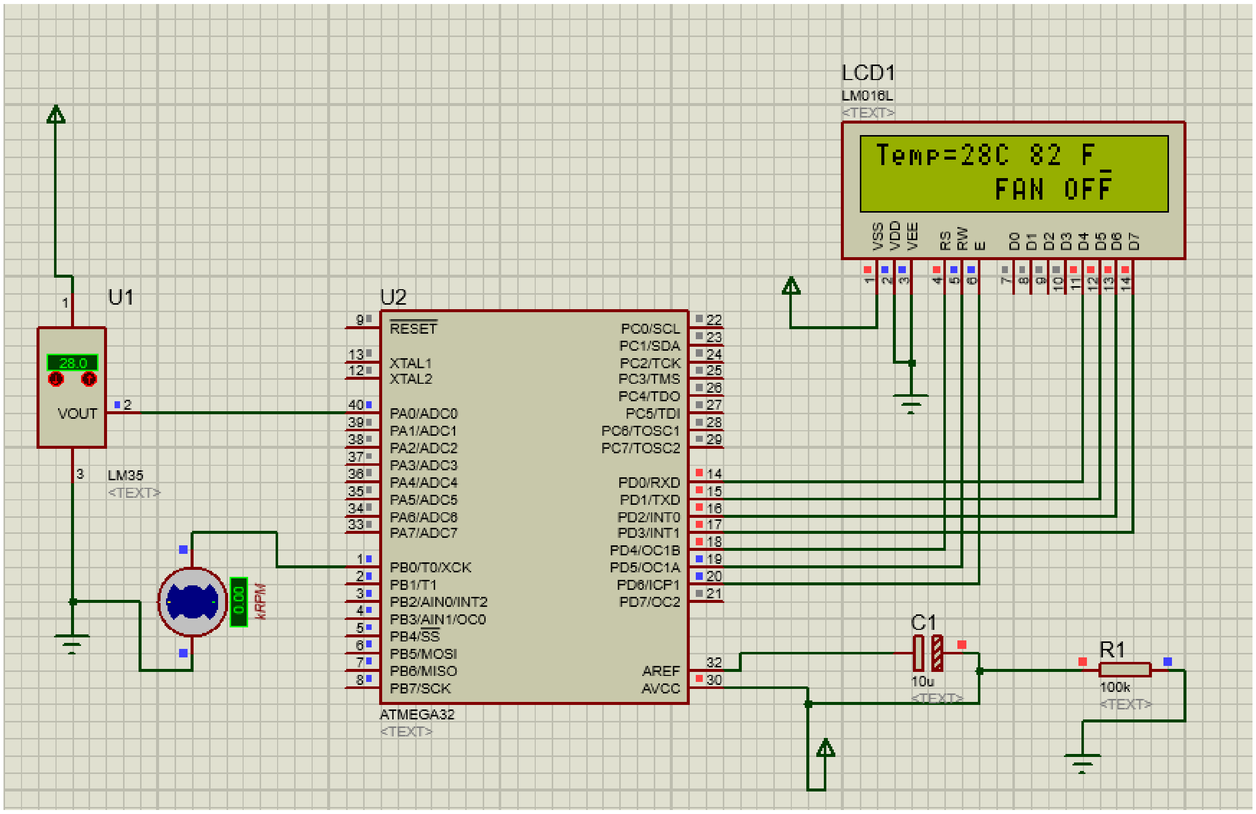
```
Embedded Systems
Exp No. : 8_
```

9. Program for interfacing various sensors and displaying quantity on LCD.

```
#ifndef F_CPU
#define F_CPU 1600000UL
#endif
#include <avr/io.h>
#include <util/delay.h>

#include "LCD/lcd.h"

void adc_init()
{
        // AREF = AVcc
        ADMUX = (1<<REFS0);

        // ADC Enable and prescaler of 128

        ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
}

// read adc value
uint16_t adc_read(uint8_t ch)
{
        // select the corresponding channel 0~7

        ch &= 0b00000111;  // AND operation with 7
        ADMUX = (ADMUX & 0xF8)|ch;

        // start single conversion
        // write '1' to ADSC
        ADCSRA |= (1<<ADSC);

        // wait for conversion to complete
        // ADSC becomes '0' again

        while(ADCSRA & (1<<ADSC));

        return (ADC);
}

int main()
{
        DDRB=0xff;
        uint16_t adc_result0;
        int temp;
        int far;
        char buffer[10];


        // initialize adc and lcd
        adc_init();
        lcd_init(LCD_DISP_ON_CURSOR); //CURSOR


        lcd_clrscr();
```

```
    lcd_gotoxy(0,0);


    _delay_ms(50);

    while(1)
    {
        adc_result0 = adc_read(0);      // read adc value at PA0

        temp=adc_result0/2.01;   // finding the temperature



        //lcd_gotoxy(0,0);
        //lcd_puts("Adc=");
        //itoa(adc_result0,buffer,10);   //display ADC value
        //lcd_puts(buffer);

        lcd_gotoxy(0,0);
        itoa(temp,buffer,10);
        lcd_puts("Temp=");   //display temperature
        lcd_puts(buffer);
        lcd_gotoxy(7,0);
        lcd_puts("C");
        far=(1.8*temp)+32;
        lcd_gotoxy(9,0);
        itoa(far,buffer,10);
        lcd_puts(buffer);
        lcd_gotoxy(12,0);
        lcd_puts("F");
        _delay_ms(1000);

        if(temp>=30)
        {lcd_clrscr();
            lcd_home();
                lcd_gotoxy(0,1);
                lcd_puts("FAN ON");

            PORTB=(1<<PINB0);

        }
        if (temp<=30)
        {
            lcd_clrscr();
            lcd_home();
            lcd_gotoxy(7,1);
            lcd_puts("FAN OFF");

            PORTB=(0<<PINB0);

        }
    }
}
```

Output:



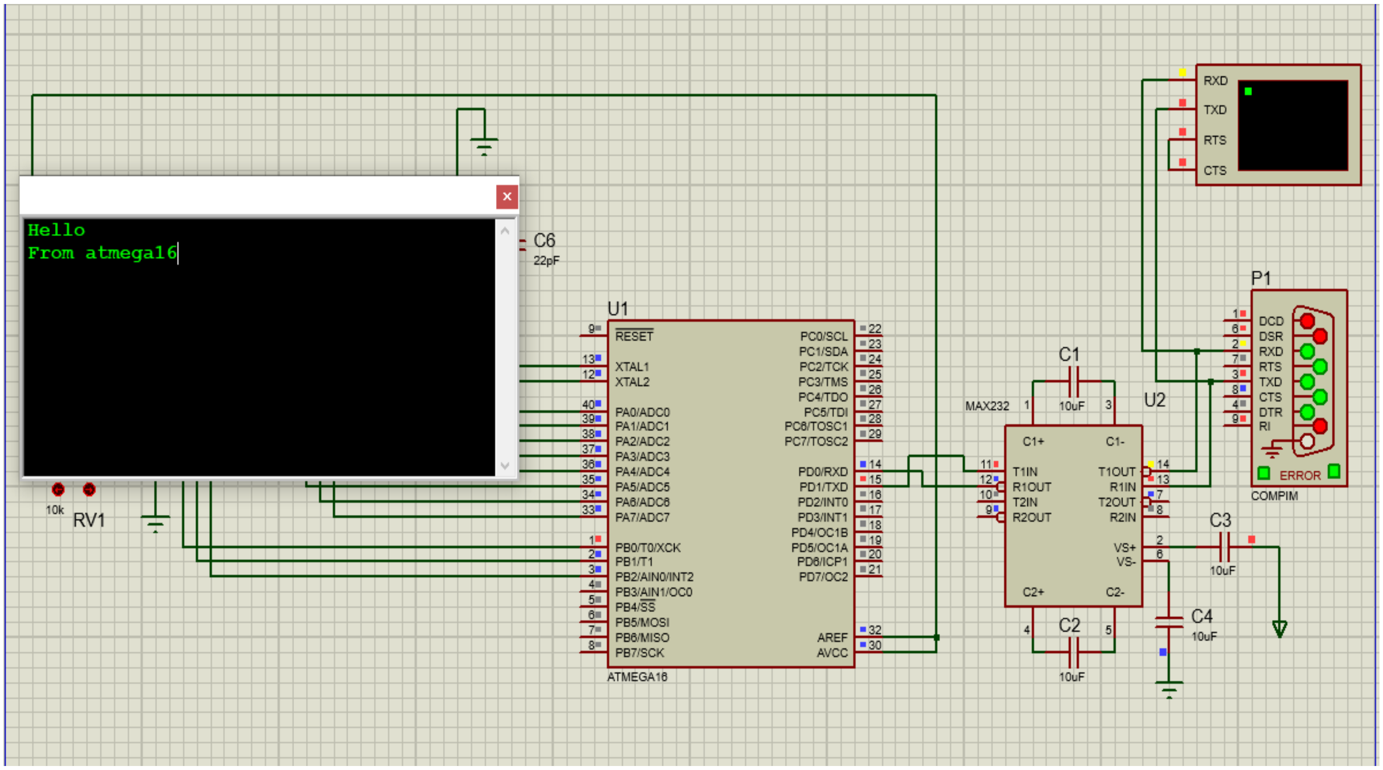
10. Program for interfacing RS 232 serial modules and file transfer using it. Using of software like terminal and hyper-terminal.

```
// Program to receive data from USART and displaying it on LCD
/*
Receive data from serial port and display it on LCD
LCD DATA port----PORT A
ctrl port------PORT B
rs-------PB0
rw-------PB1
en-------PB2
using external clock frequency 12MHz
*/

#define F_CPU 8000000UL
#define USART_BAUDRATE 9600 // Baud Rate value
#define BAUD_PRESCALE (((F_CPU / (USART_BAUDRATE * 16UL))) - 1)

#include<avr/io.h>
#include<util/delay.h>

#define LCD_DATA PORTA //LCD data port
```

```
#define ctrl PORTB
#define en PB2 // enable signal
#define rw PB1 // read/write signal
#define rs PB0 // register select signal

void LCD_cmd(unsigned char cmd);
void init_LCD(void);
void LCD_write(unsigned char data);
void LCD_clear();

void usart_init();
void usart_putch(unsigned char send);
unsigned int usart_getch();

int main()
{
        unsigned char value;
        DDRA=0xff; // LCD_DATA port as output port
        DDRB=0x07; // signal as out put
        init_LCD(); //initialization of LCD
        _delay_ms(50); // delay of 50 milli seconds
        usart_init(); // initialization of USART
        while(1)
        {
                value=usart_getch(); // Call a function to get data from serial port
                LCD_cmd(0xC0); // to go in second line and zeroth  position on LCD
                LCD_write(value); // write data to LCD
        }
        return 0;
}

void init_LCD(void)
{
        LCD_cmd(0x38); // initialization of 16X2 LCD in 8bit mode
        _delay_ms(1);

        LCD_cmd(0x01); // clear LCD
        _delay_ms(1);

        LCD_cmd(0x0E); // cursor ON
        _delay_ms(1);

        LCD_cmd(0x80); // ---8 go to first line and --0 is for 0th position
        _delay_ms(1);
        return;
}

void LCD_cmd(unsigned char cmd)
{
        LCD_DATA=cmd;
        ctrl =(0<<rs)|(0<<rw)|(1<<en);
        _delay_ms(1);
        ctrl =(0<<rs)|(0<<rw)|(0<<en);
        _delay_ms(50);
        return;
}
```

```
void LCD_write(unsigned char data)
{
        LCD_DATA= data;
        ctrl = (1<<rs)|(0<<rw)|(1<<en);
        _delay_ms(1);
        ctrl = (1<<rs)|(0<<rw)|(0<<en);
        _delay_ms(50);
        return ;
}

void usart_init()
{
        UCSRB |= (1 << RXEN) | (1 << TXEN);   // Turn on the transmission and reception 
circuitry
        UCSRC |= (1 << URSEL) | (1<<USBS) | (1 << UCSZ0) | (1 << UCSZ1); 
        // Use 8-bit character sizes

        UBRRL = BAUD_PRESCALE; 
        // Load lower 8-bits of the baud rate value into the low byte of the UBRR register
        UBRRH = (BAUD_PRESCALE >> 8); // Load upper 8-bits of the baud rate value..
        // into the high byte of the UBRR register
}

unsigned int usart_getch()
{
        while ((UCSRA & (1 << RXC)) == 0); 
        // Do nothing until data has been received and is ready to be read from UDR
        return(UDR); // return the byte
}
```

Output:



```
Hello
From atmega16
```