**ASSIGNMENT NO-1.1**

**CODE**

//LED Flashing

#include <at89c51xd2.h>

void my\_delay()

{

int i;

for(i=0;i<=10000;i++);

}

void main()

{

P1 = 0X00;

while(1)

{

P1 = 0XFF;

my\_delay();

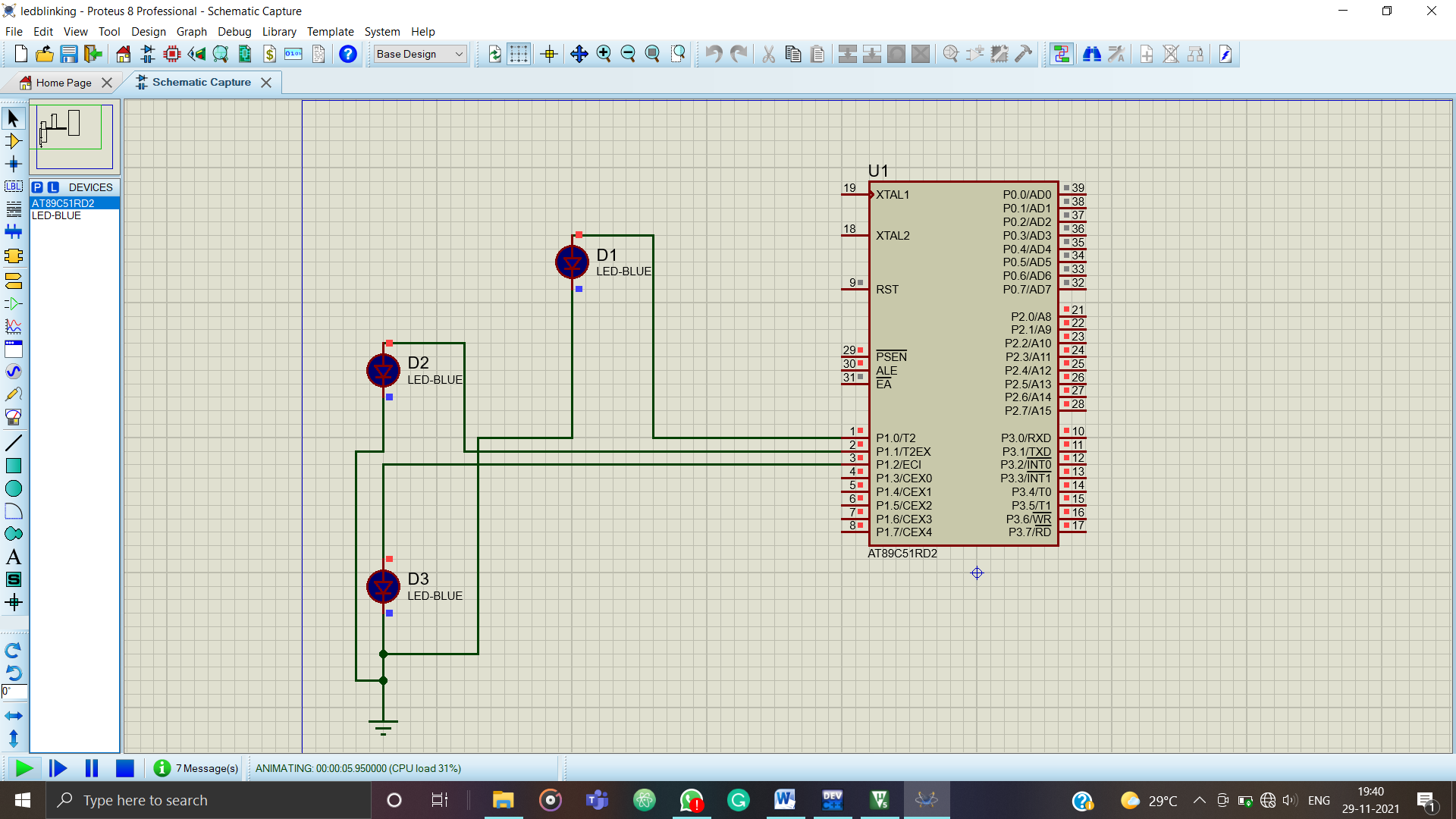
P1 = 0X00;

my\_delay();

}

}

**OUTPUT**



**ASSIGNMENT NO-1.2**

**CODE**

//Counters

#include <at89c51xd2.h>

my\_delay()

{

int i;

for(i=0;i<10000;i++);

}

void main(void)

{

P1= 0X00;

while(1)

{

unsigned char j;

for(j=0;j<255;j++)

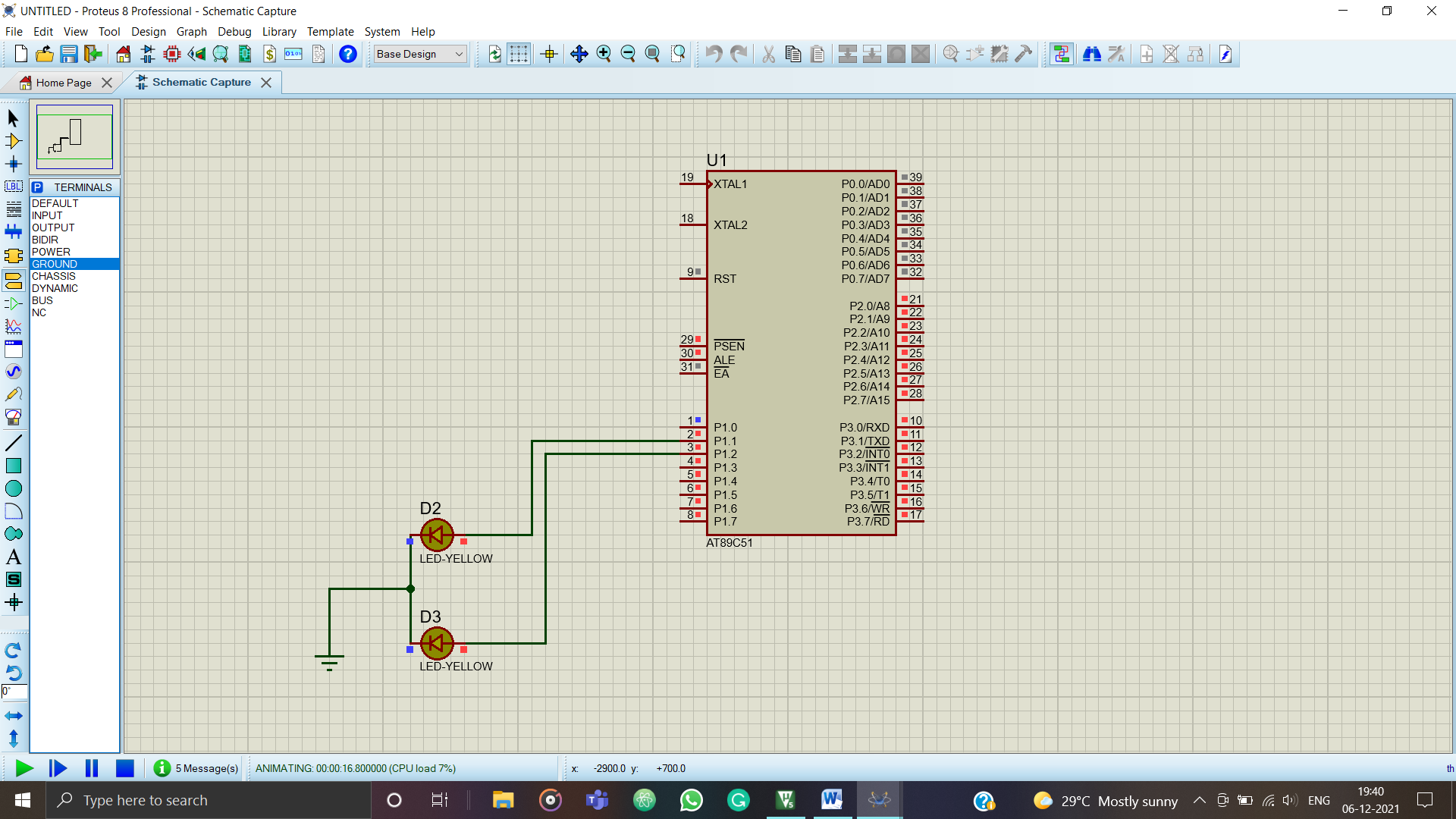
P1 = j;

my\_delay();

}

}

**OUTPUT**



**ASSIGNMENT NO-1.3**

**CODE**

**//**Display Hex Value

#include <at89c51xd2.h>

my\_delay()

{

int i;

for(i=0;i<10000;i++);

}

void main(void)

{

P1= 0X00;

while(1)

{

P1 = 0XF0; //Display binary pattern as 11110000 on Port P1

my\_delay();

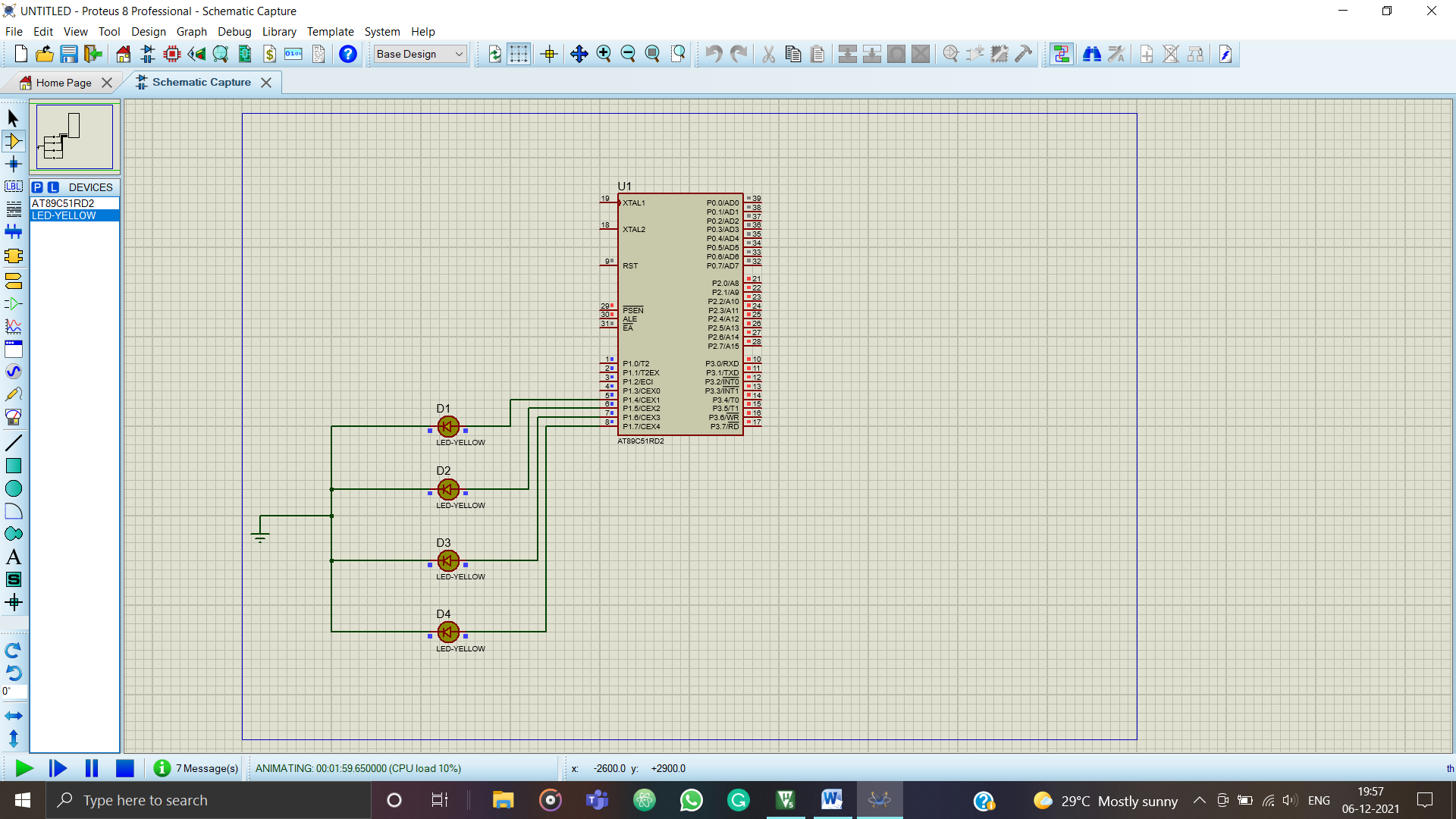
P1 = 0x00;

my\_delay();

}

}

**OUTPUT**



**ASSIGNMENT NO-2**

**CODE**

**//Sinewave**

#include <at89c51xd2.h>

void my\_delay()

{

int i;

for(i=0;i<10000;i++);

}

void main()

{

int sin\_val[12] = (128,192,238,255,238,192,120,64,17,0,17,64);

P1 = 0x00; //use p1 as output

while(1)

{

int i=0;

for(i=0;i<12;i++)

{

P1 = sin\_val[i];

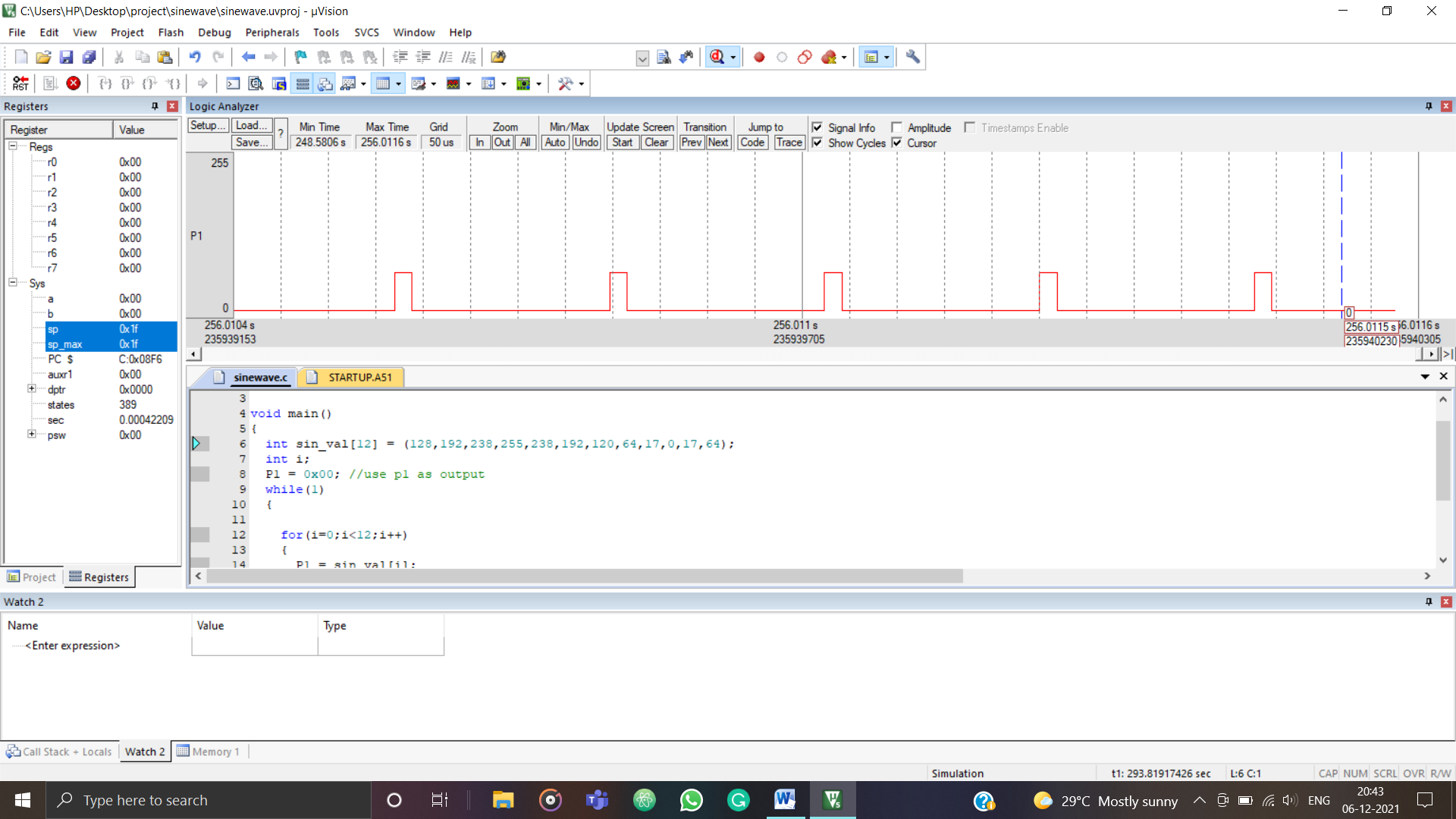
my\_delay();

}

}

}

**OUTPUT**



**//SquareWave**

#include <at89c51xd2.h>

void my\_delay()

{

int i;

for(i=0;i<10000;i++);

}

void main ()

{

int sin\_val[12] = (128,192,238,255,238,192,120,64,17,0,17,64);

P1 = 0X00; //CONVERT PORT P1 AS OUTPUT

while(1)

{

P1 = 0XFF;

my\_delay();

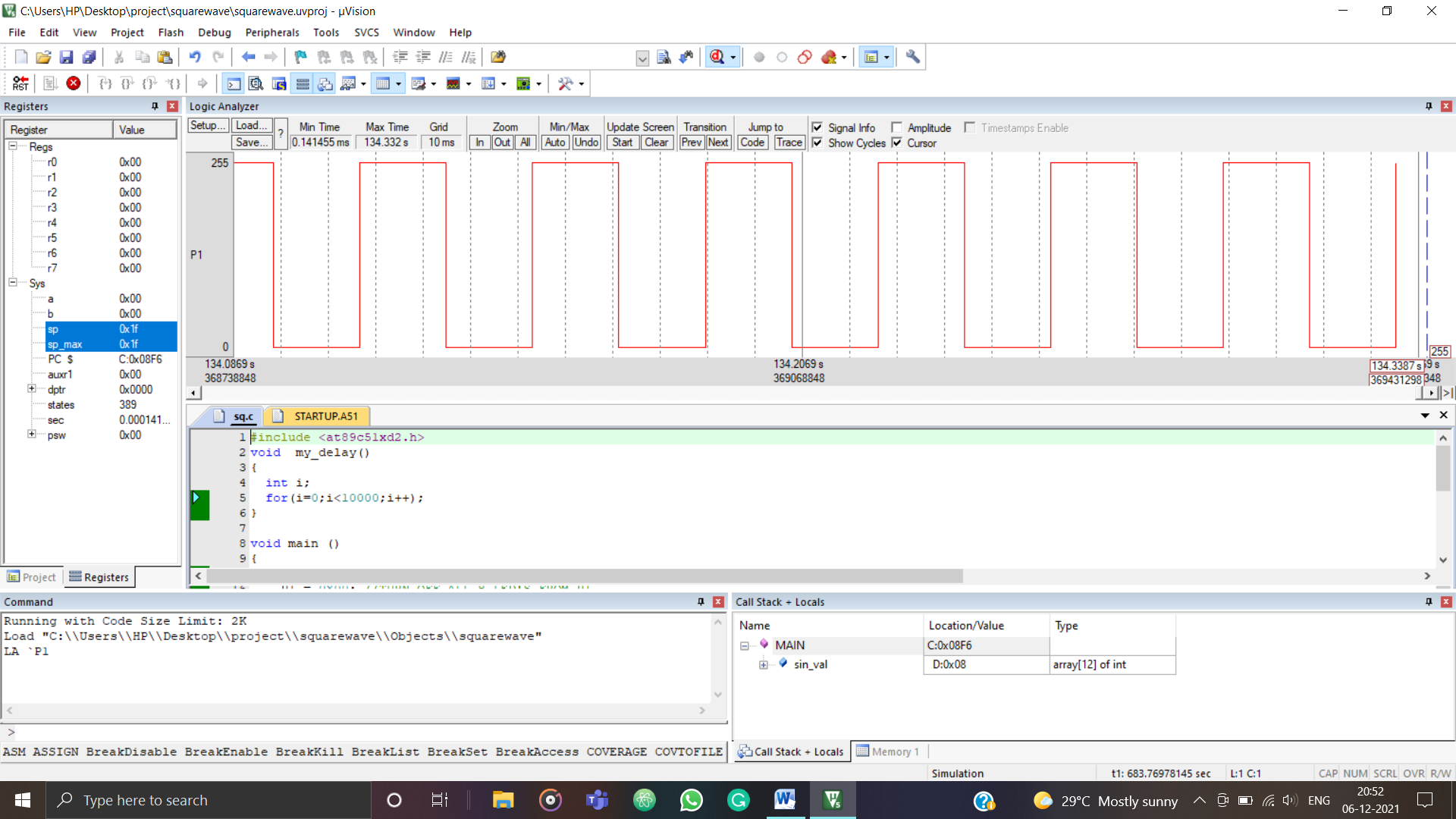
P1 = 0X00;

my\_delay();

}

}

**OUTPUT**



**ASSIGNMENT NO-3**

**CODE**

**//**7 Segment

#include <at89c51xd2.h>

#include <stdio.h>

unsigned int k[] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};

int s;

void MYDEL();

int main()

{

P1 = 0x00;

while(1)

{

for(s=0;s<10;s++)

{

P1 = k[s];

MYDEL();

MYDEL();

}

}

}

void MYDEL()

{

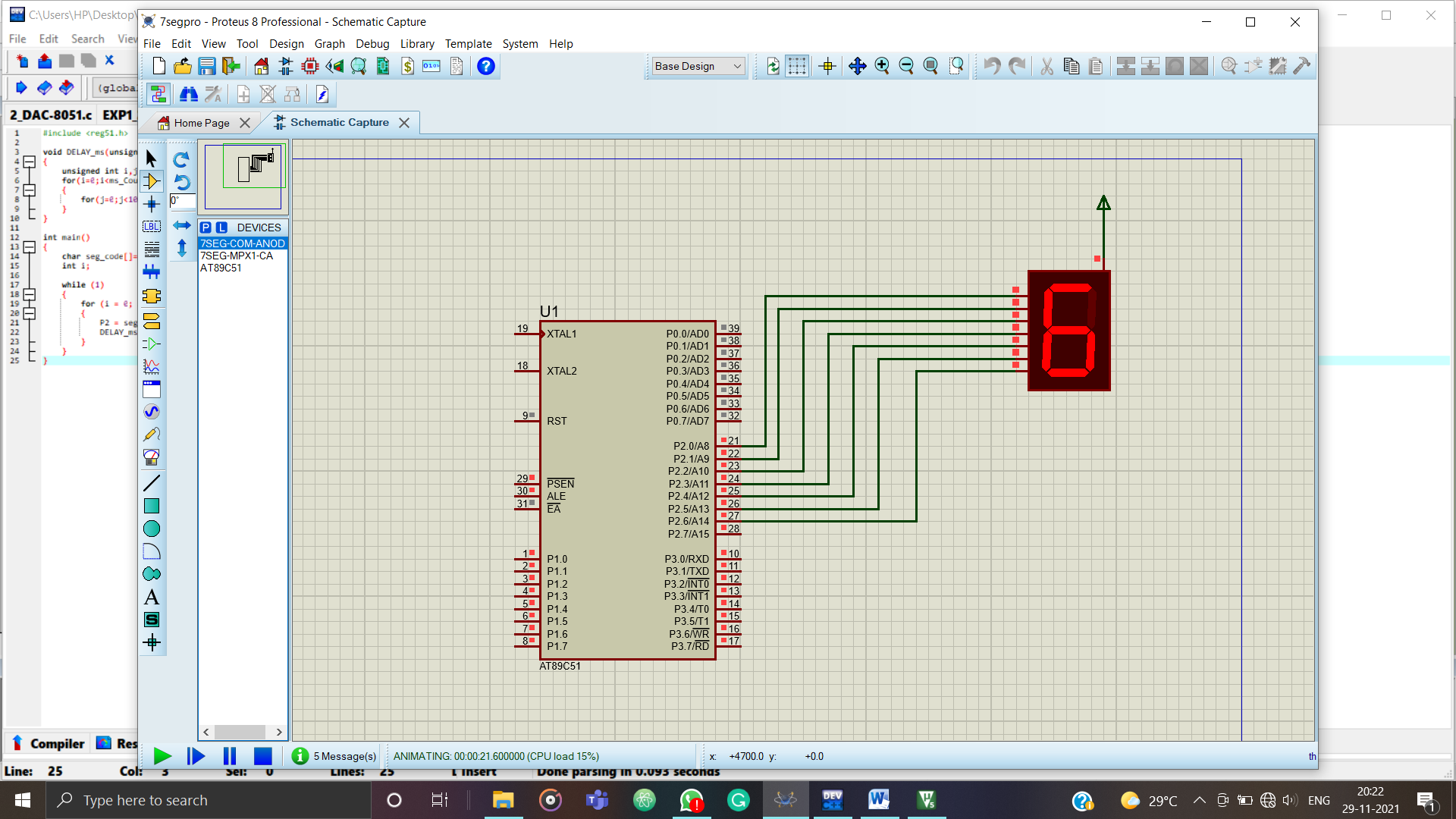
int i;

for(i=0;i<20000;i++);

return;

}

**OUTPUT**



**ASSIGNMENT NO-04**

**CODE**

#include<reg51.h>

#include<stdio.h>

void lcdinit();

void lcdcmnd(unsigned char x);

void lcddata(unsigned char y);

sbit rs=P2^0;

sbit rw=P2^1;

sbit en=P2^2;

void delay(int m, int n);

void main()

{

unsigned char t[3]="MCA";

unsigned char r[15]="MICROCONTROLLER",i;

P3=0x00;

P2=0x00;

lcdcmnd(0x38);

delay(2,2);

lcdcmnd(0x0e);

delay(2,2);

lcdcmnd(0x06);

delay(2,2);

lcdcmnd(0x085);

delay(2,2);

for(i=0;i<3;i++)

{

lcddata(t[i]);

delay(5,1);

}

lcdcmnd(0xC0);

delay(2,2);

for(i=0;i<15;i++)

{

lcddata(r[i]);

delay(5,1);

}

}

void delay(int m,int n)

{

int i,j;

for(i=0;i<m;i++)

{

for(j=0;j<n;j++)

for(i=0;i<m;i++)

{

for(j=0;j<n;j++)

{

}

}

}

}

void lcdcmnd(unsigned char x)

{

P3=x;

rs=0;

rw=0;

en=1;

delay(2,2);

en=0;

}

void lcddata(unsigned char y)

{

P3=y;

rs=1;

rw=0;

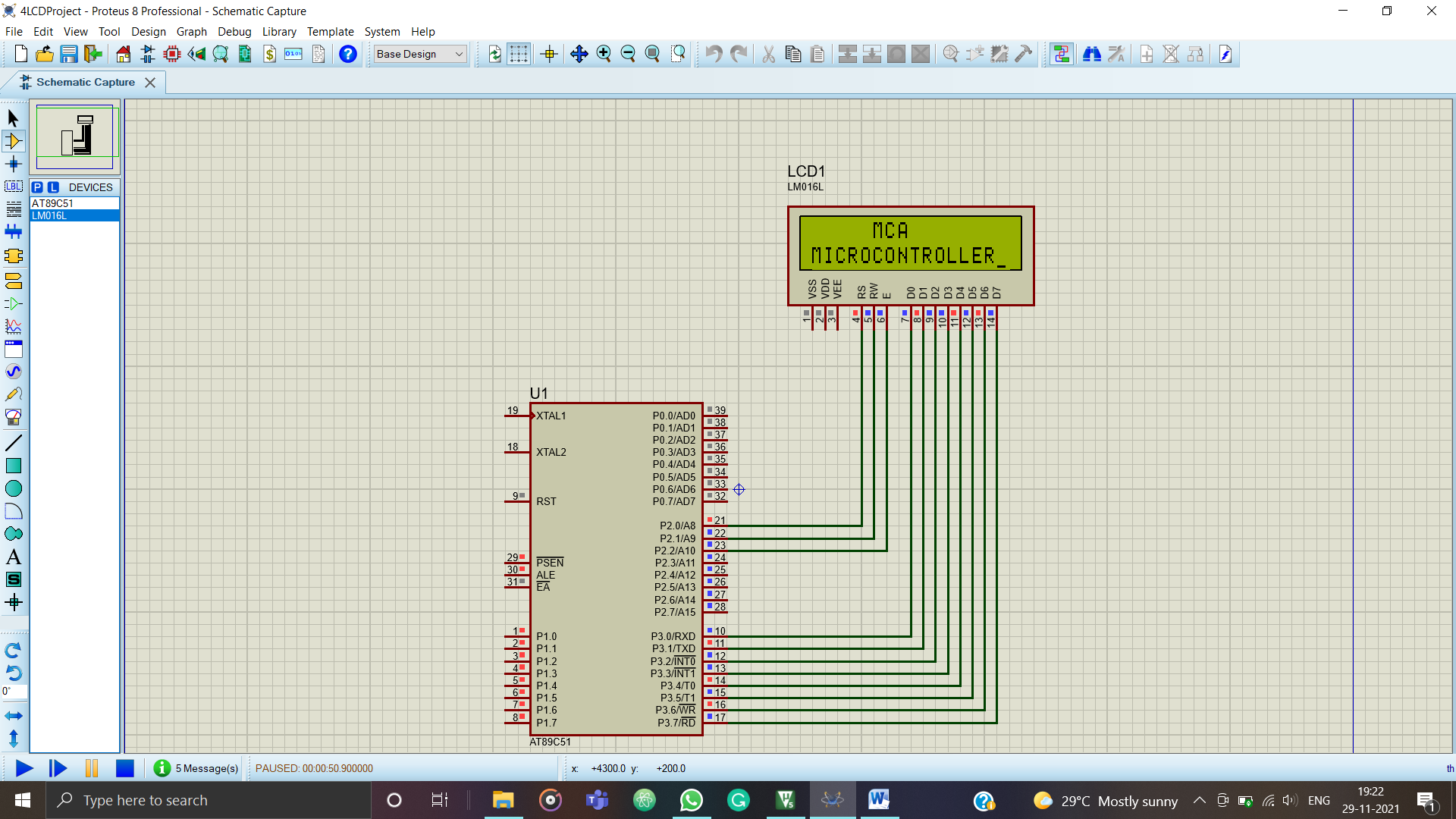
en=1;

delay(2,2);

en=0;

}

**OUTPUT**



**ASSIGNMENT NO-5**

**CODE**

// PIC18F4520 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1H

#pragma config OSC = HS // Oscillator Selection bits (HS oscillator)

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)

#pragma config IESO = OFF // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)

// CONFIG2L

#pragma config PWRT = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config BOREN = OFF // Brown-out Reset Enable bits (Brown-out Reset disabled in hardware and software)

#pragma config BORV = 3 // Brown Out Reset Voltage bits (Minimum setting)

// CONFIG2H

#pragma config WDT = OFF // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))

#pragma config WDTPS = 32768 // Watchdog Timer Postscale Select bits (1:32768)

// CONFIG3H

#pragma config CCP2MX = PORTC // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)

#pragma config PBADEN = OFF // PORTB A/D Enable bit (PORTB<4:0> pins are configured as digital I/O on Reset)

#pragma config LPT1OSC = OFF // Low-Power Timer1 Oscillator Enable bit (Timer1 configured for higher power operation)

#pragma config MCLRE = OFF // MCLR Pin Enable bit (RE3 input pin enabled; MCLR disabled)

// CONFIG4L

#pragma config STVREN = OFF // Stack Full/Underflow Reset Enable bit (Stack full/underflow will not cause Reset)

#pragma config LVP = OFF // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)

#pragma config XINST = OFF // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode))

// CONFIG5L

#pragma config CP0 = OFF // Code Protection bit (Block 0 (000800-001FFFh) not code-protected)

#pragma config CP1 = OFF // Code Protection bit (Block 1 (002000-003FFFh) not code-protected)

#pragma config CP2 = OFF // Code Protection bit (Block 2 (004000-005FFFh) not code-protected)

#pragma config CP3 = OFF // Code Protection bit (Block 3 (006000-007FFFh) not code-protected)

// CONFIG5H

#pragma config CPB = OFF // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected)

#pragma config CPD = OFF // Data EEPROM Code Protection bit (Data EEPROM not code-protected)

// CONFIG6L

#pragma config WRT0 = OFF // Write Protection bit (Block 0 (000800-001FFFh) not write-protected)

#pragma config WRT1 = OFF // Write Protection bit (Block 1 (002000-003FFFh) not write-protected)

#pragma config WRT2 = OFF // Write Protection bit (Block 2 (004000-005FFFh) not write-protected)

#pragma config WRT3 = OFF // Write Protection bit (Block 3 (006000-007FFFh) not write-protected)

// CONFIG6H

#pragma config WRTC = OFF // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write-protected)

#pragma config WRTB = OFF // Boot Block Write Protection bit (Boot block (000000-0007FFh) not write-protected)

#pragma config WRTD = OFF // Data EEPROM Write Protection bit (Data EEPROM not write-protected)

// CONFIG7L

#pragma config EBTR0 = OFF // Table Read Protection bit (Block 0 (000800-001FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR1 = OFF // Table Read Protection bit (Block 1 (002000-003FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR2 = OFF // Table Read Protection bit (Block 2 (004000-005FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR3 = OFF // Table Read Protection bit (Block 3 (006000-007FFFh) not protected from table reads executed in other blocks)

// CONFIG7H

#pragma config EBTRB = OFF // Boot Block Table Read Protection bit (Boot block (000000-0007FFh) not protected from table reads executed in other blocks)

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

#include <xc.h>

/\*Start of main program\*/

void msdelay (unsigned int itime);

void Right(void);

void Left(void);

#define SW2 PORTBbits.RB0

#define SW1 PORTBbits.RB1

#define relay PORTBbits.RB3

#define D1 PORTBbits.RB4

#define D2 PORTBbits.RB5

#define D3 PORTBbits.RB6

#define D4 PORTBbits.RB7

void main()

{

ADCON1=0X0F; //no need of ADC CONVERSION,OFF

TRISBbits.TRISB0=1;

TRISBbits.TRISB1=1;

TRISBbits.TRISB3=0;

TRISBbits.TRISB4=0;

TRISBbits.TRISB5=0;

TRISBbits.TRISB6=0;

TRISBbits.TRISB7=0;

D1=D2=D3=D4=0;//INITIALLY ALL LED'S ARE IN OFF POSITION

SW1=SW2=1;

relay=0;

while(1)

{

if(SW2==0 & SW1==1)

{

relay=1;

Right();

}

if(SW1==0 & SW2==1 )

{

relay=0;

Left();

}

}

}

void Right(void)

{

D1=D2=D3=D4=0;

while(SW1!=0)

{

relay=1;

D1=1;D2=0;D3=0;D4=0;

msdelay(10);

D1=0;D2=1;D3=0;D4=0;

msdelay(10);

D1=0;D2=0;D3=1;D4=0;

msdelay(10);

D1=0;D2=0;D3=0;D4=1;

msdelay(10);

}

}

void Left(void)

{

D1=D2=D3=D4=0;

while(SW2!=0)

{

relay=0;

D1=0;D2=0;D3=0;D4=1;

msdelay(10);

D1=0;D2=0;D3=1;D4=0;

msdelay(10);

D1=0;D2=1;D3=0;D4=0;

msdelay(10);

D1=1;D2=0;D3=0;D4=0;

msdelay(10);

}

}

void msdelay(unsigned int itime)

{

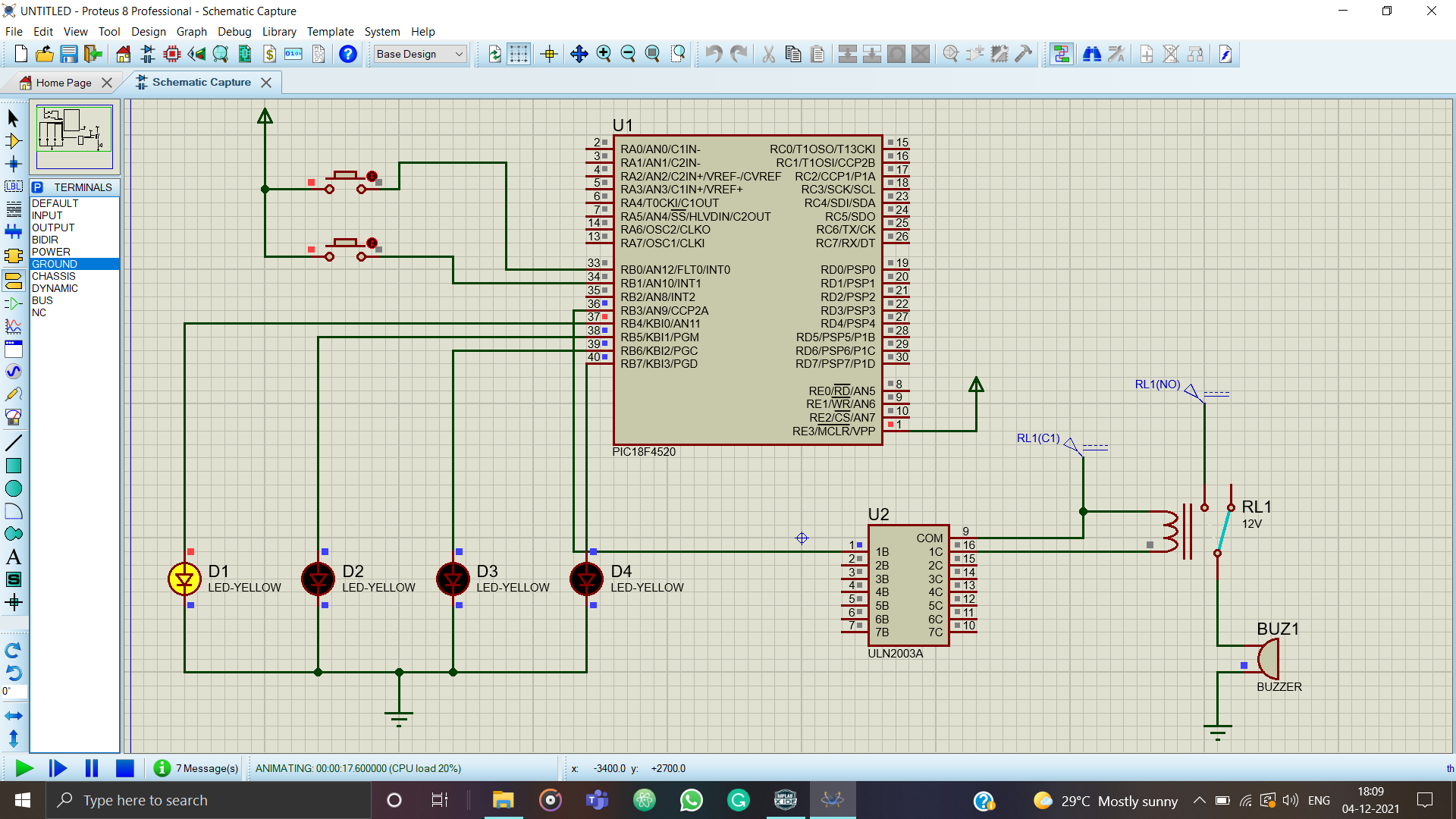
int i,j;

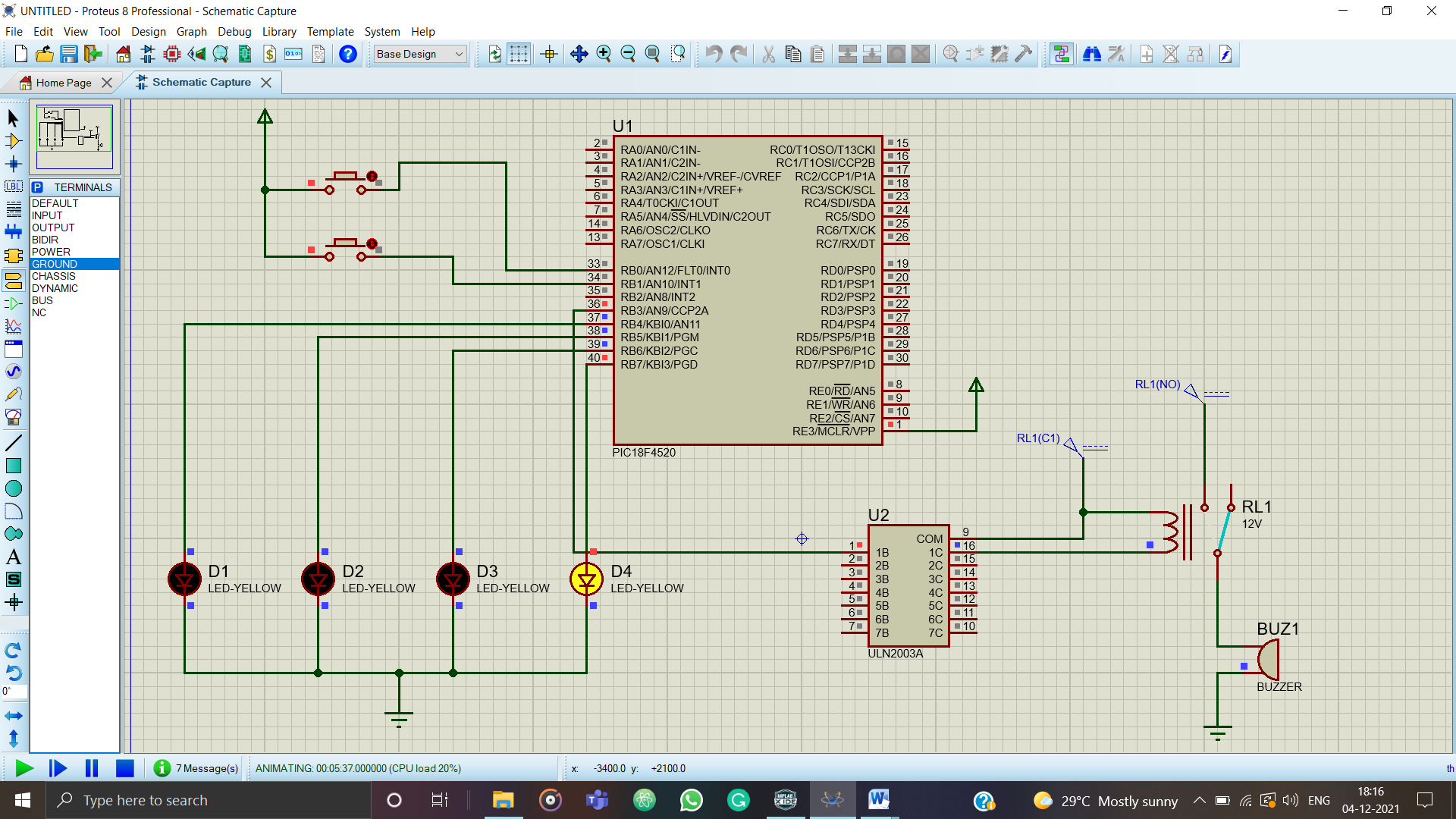
for(i=0;i<itime;i++)

for(j=0;j<1275;j++);

}

**OUTPUT**





**ASSIGNMENT NO-6**

**CODE**

// PIC18F4520 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1H

#pragma config OSC = HS // Oscillator Selection bits (HS oscillator)

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)

#pragma config IESO = OFF // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)

// CONFIG2L

#pragma config PWRT = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config BOREN = OFF // Brown-out Reset Enable bits (Brown-out Reset disabled in hardware and software)

#pragma config BORV = 3 // Brown Out Reset Voltage bits (Minimum setting)

// CONFIG2H

#pragma config WDT = OFF // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))

#pragma config WDTPS = 32768 // Watchdog Timer Postscale Select bits (1:32768)

// CONFIG3H

#pragma config CCP2MX = PORTC // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)

#pragma config PBADEN = OFF // PORTB A/D Enable bit (PORTB<4:0> pins are configured as digital I/O on Reset)

#pragma config LPT1OSC = OFF // Low-Power Timer1 Oscillator Enable bit (Timer1 configured for higher power operation)

#pragma config MCLRE = OFF // MCLR Pin Enable bit (RE3 input pin enabled; MCLR disabled)

// CONFIG4L

#pragma config STVREN = OFF // Stack Full/Underflow Reset Enable bit (Stack full/underflow will not cause Reset)

#pragma config LVP = OFF // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)

#pragma config XINST = OFF // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode))

// CONFIG5L

#pragma config CP0 = OFF // Code Protection bit (Block 0 (000800-001FFFh) not code-protected)

#pragma config CP1 = OFF // Code Protection bit (Block 1 (002000-003FFFh) not code-protected)

#pragma config CP2 = OFF // Code Protection bit (Block 2 (004000-005FFFh) not code-protected)

#pragma config CP3 = OFF // Code Protection bit (Block 3 (006000-007FFFh) not code-protected)

// CONFIG5H

#pragma config CPB = OFF // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected)

#pragma config CPD = OFF // Data EEPROM Code Protection bit (Data EEPROM not code-protected)

// CONFIG6L

#pragma config WRT0 = OFF // Write Protection bit (Block 0 (000800-001FFFh) not write-protected)

#pragma config WRT1 = OFF // Write Protection bit (Block 1 (002000-003FFFh) not write-protected)

#pragma config WRT2 = OFF // Write Protection bit (Block 2 (004000-005FFFh) not write-protected)

#pragma config WRT3 = OFF // Write Protection bit (Block 3 (006000-007FFFh) not write-protected)

// CONFIG6H

#pragma config WRTC = OFF // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write-protected)

#pragma config WRTB = OFF // Boot Block Write Protection bit (Boot block (000000-0007FFh) not write-protected)

#pragma config WRTD = OFF // Data EEPROM Write Protection bit (Data EEPROM not write-protected)

// CONFIG7L

#pragma config EBTR0 = OFF // Table Read Protection bit (Block 0 (000800-001FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR1 = OFF // Table Read Protection bit (Block 1 (002000-003FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR2 = OFF // Table Read Protection bit (Block 2 (004000-005FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR3 = OFF // Table Read Protection bit (Block 3 (006000-007FFFh) not protected from table reads executed in other blocks)

// CONFIG7H

#pragma config EBTRB = OFF // Boot Block Table Read Protection bit (Boot block (000000-0007FFh) not protected from table reads executed in other blocks)

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

#include <xc.h>

/\*Start of main program\*/

void interrupt \_timer\_isr(void)

{

TMR0H = 0X48;

TMR0L = 0XE5;

PORTB = ~PORTB; //Toggle the PORTB led outputs RB0 - RB3

INTCONbits.TMR0IF = 0; //Resetting the timer overflow interrupt flag

return;

}

void main()

{

ADCON1 = 0x0F; //Configuring the PORTE pins as digital I/O

TRISB = 0; //Configruing the LED port pins as outputs

PORTB = 0xFF; //Setting the initial value of the LED's after reset

T0CON = 0x04;

/\*Set the timer to 16-bit mode,internal instruction cycle clock and prescaler\*/

TMR0H = 0x48; // Reset Timer0 to 0x48E5 TO MAKE DELAY

TMR0L = 0xE5;

INTCONbits.TMR0IF = 0; // Clear Timer0 overflow flag

INTCONbits.TMR0IE = 1; // TMR0 interrupt enabled

T0CONbits.TMR0ON = 1; // Start timer0

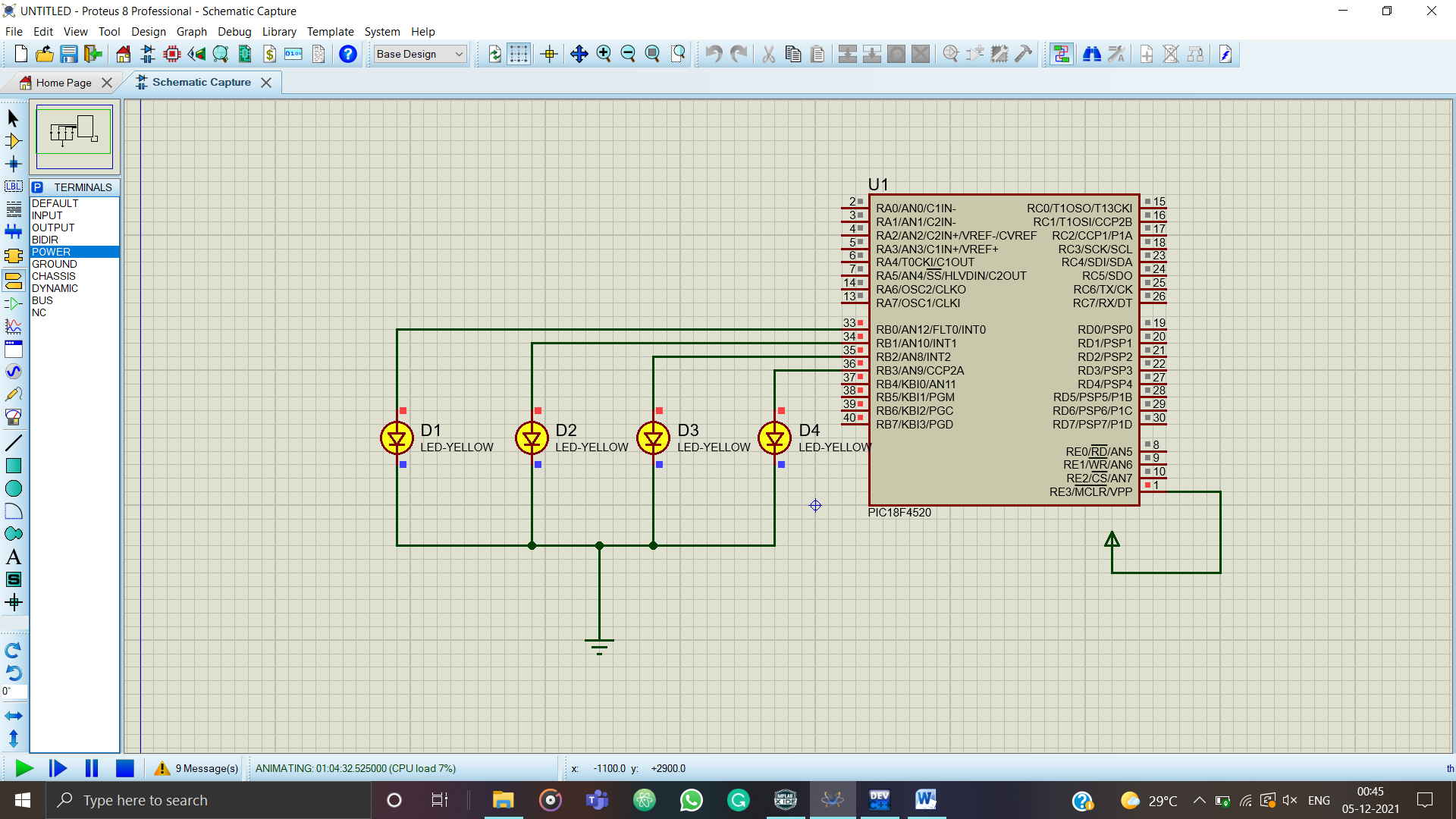
INTCONbits.GIE = 1; // Global interrupt enabled

while(1);

/\*Program execution stays here untill the timer overflow interrupt is generated\*/

}

**OUTPUT**



**ASSIGNMENT NO-7**

**CODE**

// PIC18F4520 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1H

#pragma config OSC = HS // Oscillator Selection bits (HS oscillator)

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)

#pragma config IESO = OFF // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)

// CONFIG2L

#pragma config PWRT = ON // Power-up Timer Enable bit (PWRT enabled)

#pragma config BOREN = SBORDIS // Brown-out Reset Enable bits (Brown-out Reset enabled in hardware only (SBOREN is disabled))

#pragma config BORV = 3 // Brown Out Reset Voltage bits (Minimum setting)

// CONFIG2H

#pragma config WDT = OFF // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))

#pragma config WDTPS = 32768 // Watchdog Timer Postscale Select bits (1:32768)

// CONFIG3H

#pragma config CCP2MX = PORTC // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)

#pragma config PBADEN = ON // PORTB A/D Enable bit (PORTB<4:0> pins are configured as analog input channels on Reset)

#pragma config LPT1OSC = OFF // Low-Power Timer1 Oscillator Enable bit (Timer1 configured for higher power operation)

#pragma config MCLRE = ON // MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled)

// CONFIG4L

#pragma config STVREN = ON // Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset)

#pragma config LVP = OFF // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)

#pragma config XINST = OFF // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode))

// CONFIG5L

#pragma config CP0 = OFF // Code Protection bit (Block 0 (000800-001FFFh) not code-protected)

#pragma config CP1 = OFF // Code Protection bit (Block 1 (002000-003FFFh) not code-protected)

#pragma config CP2 = OFF // Code Protection bit (Block 2 (004000-005FFFh) not code-protected)

#pragma config CP3 = OFF // Code Protection bit (Block 3 (006000-007FFFh) not code-protected)

// CONFIG5H

#pragma config CPB = OFF // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected)

#pragma config CPD = OFF // Data EEPROM Code Protection bit (Data EEPROM not code-protected)

// CONFIG6L

#pragma config WRT0 = OFF // Write Protection bit (Block 0 (000800-001FFFh) not write-protected)

#pragma config WRT1 = OFF // Write Protection bit (Block 1 (002000-003FFFh) not write-protected)

#pragma config WRT2 = OFF // Write Protection bit (Block 2 (004000-005FFFh) not write-protected)

#pragma config WRT3 = OFF // Write Protection bit (Block 3 (006000-007FFFh) not write-protected)

// CONFIG6H

#pragma config WRTC = OFF // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write-protected)

#pragma config WRTB = OFF // Boot Block Write Protection bit (Boot block (000000-0007FFh) not write-protected)

#pragma config WRTD = OFF // Data EEPROM Write Protection bit (Data EEPROM not write-protected)

// CONFIG7L

#pragma config EBTR0 = OFF // Table Read Protection bit (Block 0 (000800-001FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR1 = OFF // Table Read Protection bit (Block 1 (002000-003FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR2 = OFF // Table Read Protection bit (Block 2 (004000-005FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR3 = OFF // Table Read Protection bit (Block 3 (006000-007FFFh) not protected from table reads executed in other blocks)

// CONFIG7H

#pragma config EBTRB = OFF // Boot Block Table Read Protection bit (Boot block (000000-0007FFh) not protected from table reads executed in other blocks)

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

#include <xc.h>

#define rs LATBbits.LB0

#define rw LATBbits.LB1

#define en LATBbits.LB2

//LCD Data pins

#define lcdport LATC

#define \_XTAL\_FREQ 20000000

void lcd\_ini();

void lcdcmd(unsigned char);

void lcddata(unsigned char);

void adc\_con(unsigned int);

void adc\_init();

void Delay\_ms(int mtime);

unsigned char data[20]="ADC OUTPUT=";

unsigned int digital\_out,avg\_output=0,temp;

unsigned int i=0;

void main()

{

TRISA=0x01; // Configure RA0 as input pin

// LATA=0;

TRISB=0; // Configure Port B as output port

LATB=0;

TRISC=0;

LATC=0;

lcd\_ini(); // LCD initialization

while(data[i]!='\0')

{

lcddata(data[i]); // Call lcddata function to send character one by from 'data' array

i++;

}

adc\_init(); //ADC Initialization while(1)

{

temp=0;

ADCON0bits.GO = 1;

while(ADCON0bits.DONE == 1); // Wait until Processing of AD in finished

digital\_out=((ADRESL)|(ADRESH<<8)); // Store 10-bit output into a 16-bit variable

Delay\_ms(20);

adc\_con(digital\_out); //Function to convert the decimal vaule to its corresponding ASCII

}

}

void adc\_init()

{

ADCON0 = 0x01; // Fosc/64 , Chan 0 , AD On

ADCON1 = 0xCE; // RIght justified , AN0 selected & Fosc/64

ADCON2 = 0xAA;

ADCON0bits.ADON = 1; // Enable ADC

}

void lcd\_ini()

{

lcdcmd(0x38); // Configure the LCD in 8-bit mode, 2 line and 5x7 font

lcdcmd(0x0C); // Display On and Cursor Off lcdcmd(0x01); // Clear display screen lcdcmd(0x06); // Increment cursor

lcdcmd(0x80); // Set cursor position to 1st line, 1st column

}

void adc\_con(unsigned int adc\_out)

{

unsigned int adc\_out1;

int i=0;

char position=0xC3;

for(i=0;i<=3;i++)

{

adc\_out1=adc\_out%10; // To exract the unit position digit

adc\_out=adc\_out/10;

lcdcmd(position);

lcddata(48+adc\_out1); // Convert into its corresponding ASCII

position--;

}

}

void lcdcmd(unsigned char cmdout)

{

lcdport=cmdout; //Send command to lcdport=PORTB

rs=0;

rw=0;

en=1; Delay\_ms(10); en=0;

}

void lcddata(unsigned char dataout)

{

lcdport=dataout; //Send data to lcdport=PORTB

rs=1;

rw=0;

en=1;

Delay\_ms(10);

en=0;

}

void Delay\_ms(int mtime)

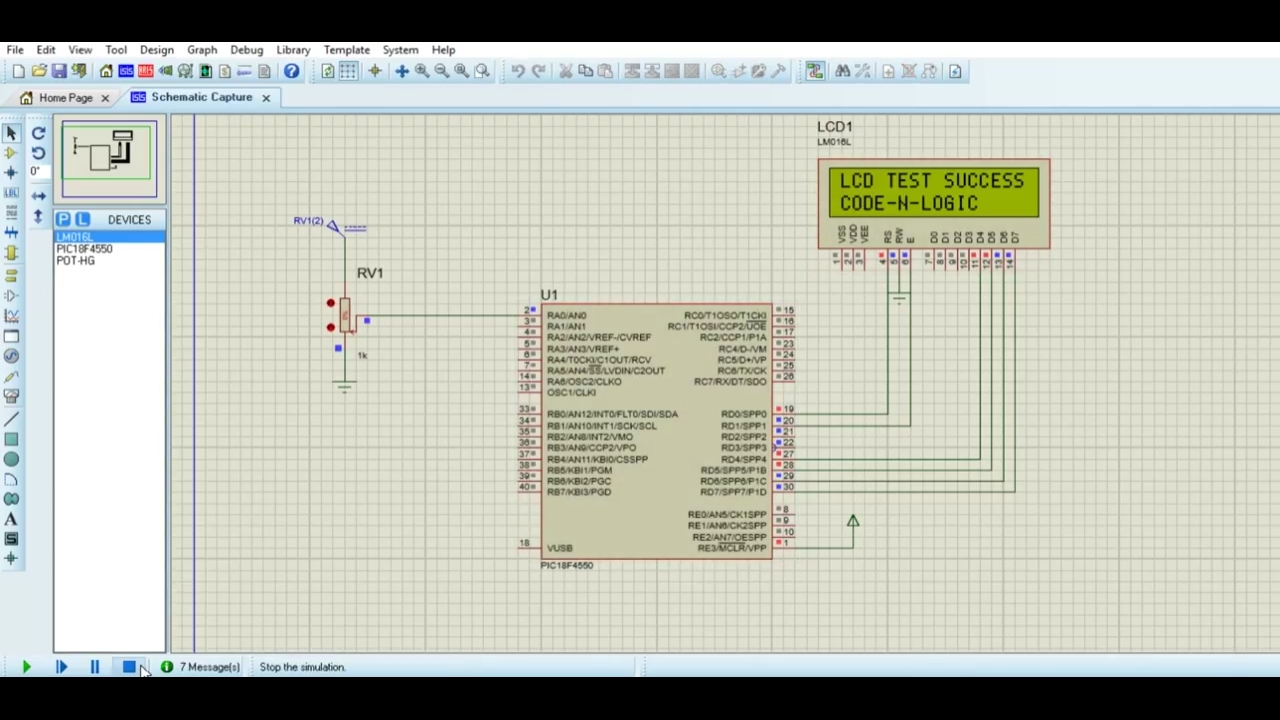
{

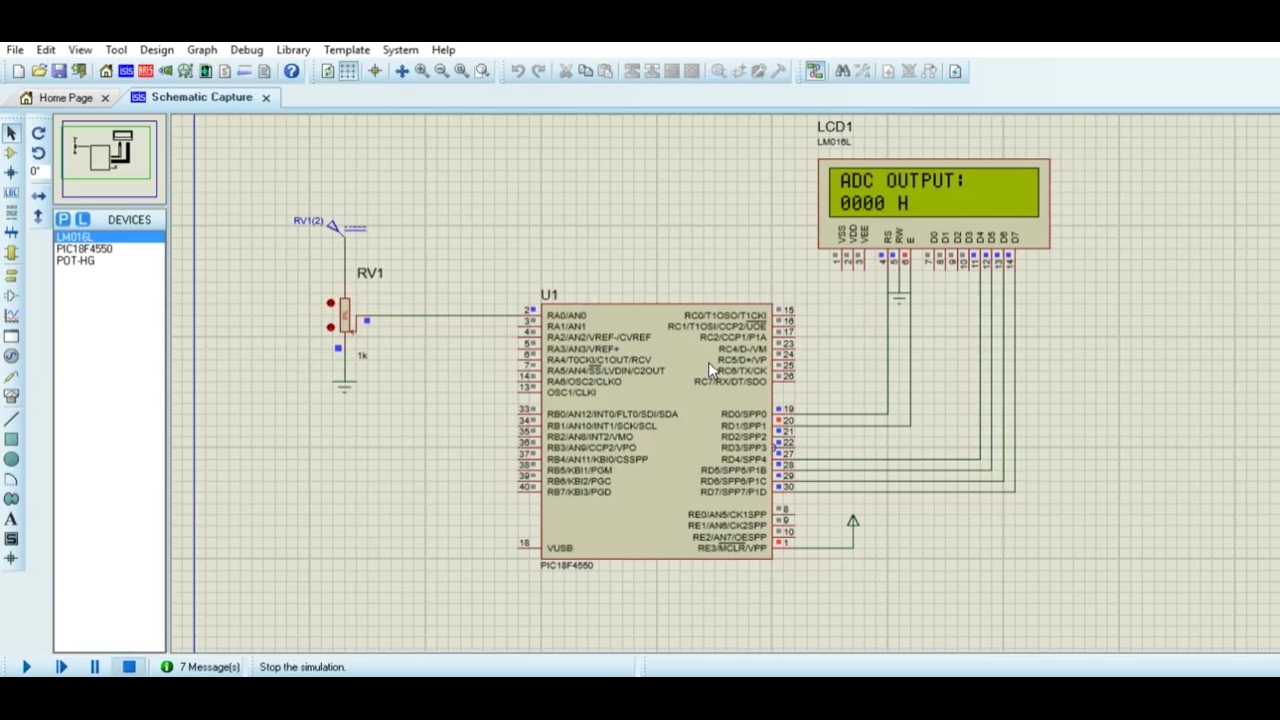
for(int i = 0; i<= mtime ; i++)

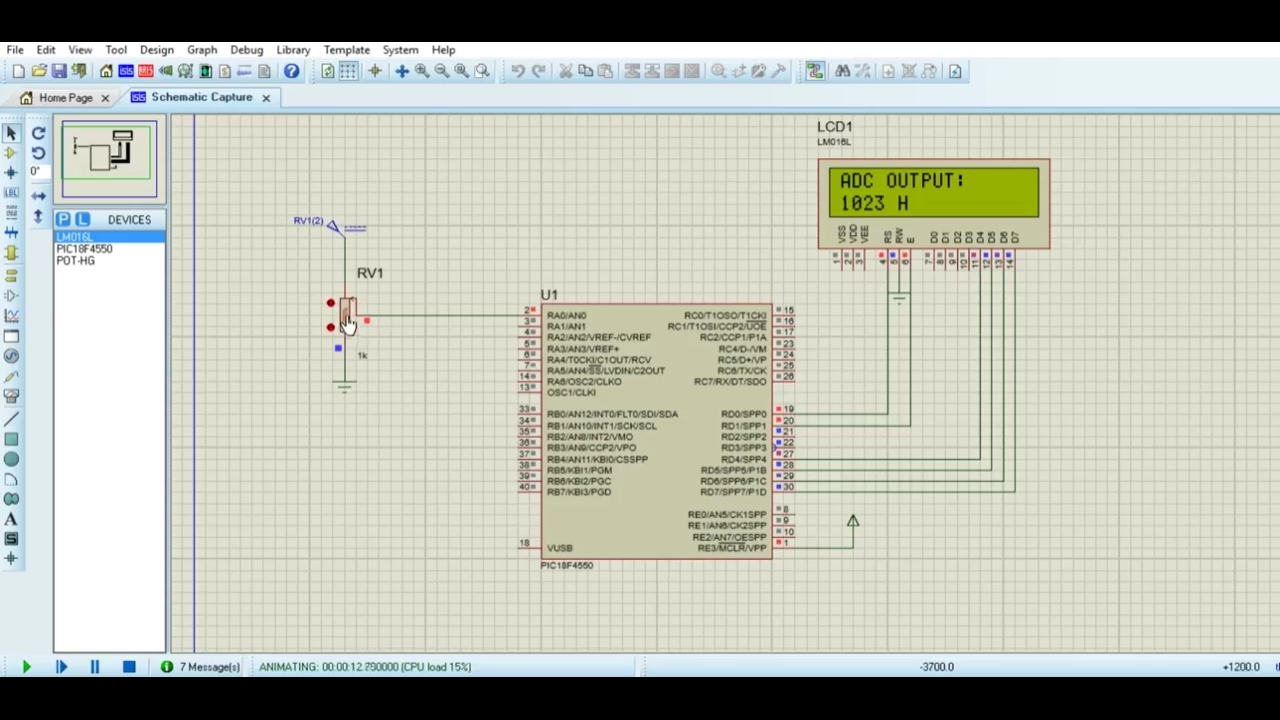
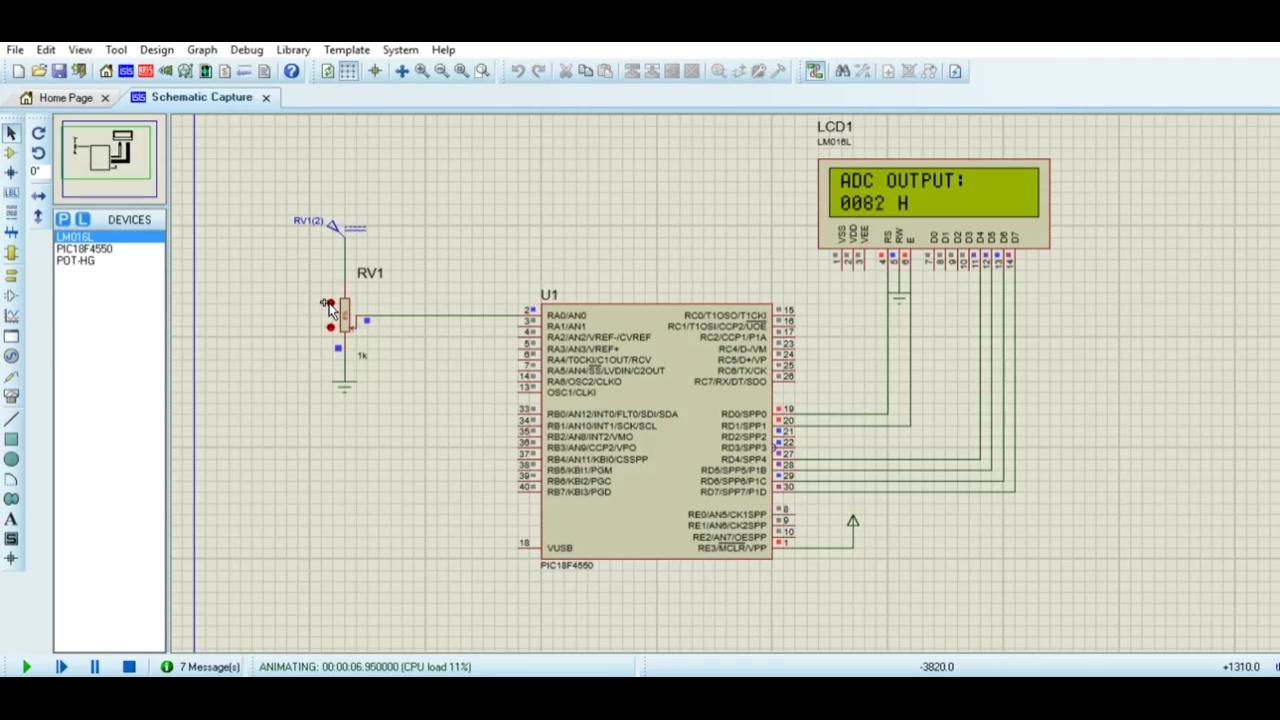
\_\_delay\_ms(1);

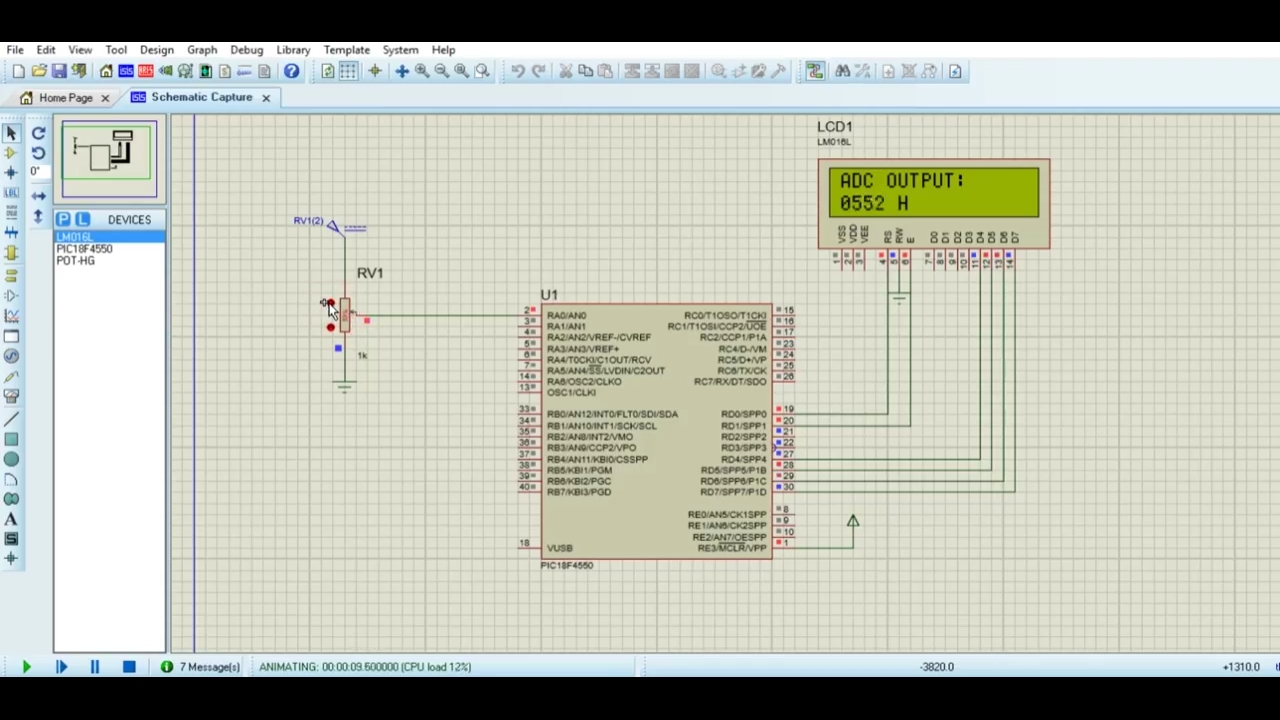
}

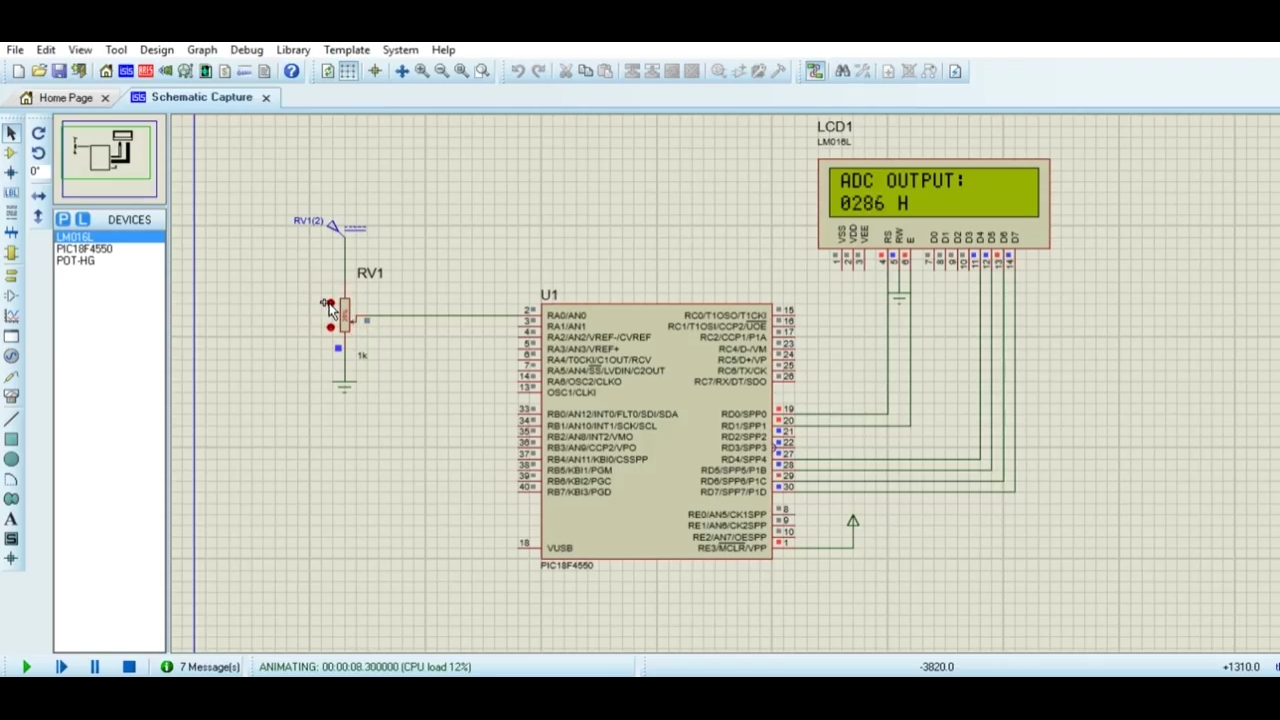
**OUTPUT**

****

****

****

****

****

**ASSIGNMENT NO-8**

**CODE**

// PIC18F4520 Configuration Bit Settings

// 'C' source line config statements

#include <xc.h>

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

// CONFIG1H

#pragma config OSC = HSPLL // Oscillator Selection bits (HS oscillator, PLL enabled (Clock Frequency = 4 x FOSC1))

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)

#pragma config IESO = OFF // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)

// CONFIG2L

#pragma config PWRT = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config BOREN = OFF // Brown-out Reset Enable bits (Brown-out Reset disabled in hardware and software)

#pragma config BORV = 3 // Brown Out Reset Voltage bits (Minimum setting)

// CONFIG2H

#pragma config WDT = OFF // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))

#pragma config WDTPS = 32768 // Watchdog Timer Postscale Select bits (1:32768)

// CONFIG3H

#pragma config CCP2MX = PORTC // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)

#pragma config PBADEN = OFF // PORTB A/D Enable bit (PORTB<4:0> pins are configured as digital I/O on Reset)

#pragma config LPT1OSC = OFF // Low-Power Timer1 Oscillator Enable bit (Timer1 configured for higher power operation)

#pragma config MCLRE = OFF // MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled)

// CONFIG4L

#pragma config STVREN = ON // Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset)

#pragma config LVP = OFF // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)

#pragma config XINST = OFF // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode))

// CONFIG5L

#pragma config CP0 = OFF // Code Protection bit (Block 0 (000800-001FFFh) not code-protected)

#pragma config CP1 = OFF // Code Protection bit (Block 1 (002000-003FFFh) not code-protected)

#pragma config CP2 = OFF // Code Protection bit (Block 2 (004000-005FFFh) not code-protected)

#pragma config CP3 = OFF // Code Protection bit (Block 3 (006000-007FFFh) not code-protected)

// CONFIG5H

#pragma config CPB = OFF // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected)

#pragma config CPD = OFF // Data EEPROM Code Protection bit (Data EEPROM not code-protected)

// CONFIG6L

#pragma config WRT0 = OFF // Write Protection bit (Block 0 (000800-001FFFh) not write-protected)

#pragma config WRT1 = OFF // Write Protection bit (Block 1 (002000-003FFFh) not write-protected)

#pragma config WRT2 = OFF // Write Protection bit (Block 2 (004000-005FFFh) not write-protected)

#pragma config WRT3 = OFF // Write Protection bit (Block 3 (006000-007FFFh) not write-protected)

// CONFIG6H

#pragma config WRTC = OFF // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write-protected)

#pragma config WRTB = OFF // Boot Block Write Protection bit (Boot block (000000-0007FFh) not write-protected)

#pragma config WRTD = OFF // Data EEPROM Write Protection bit (Data EEPROM not write-protected)

// CONFIG7L

#pragma config EBTR0 = OFF // Table Read Protection bit (Block 0 (000800-001FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR1 = OFF // Table Read Protection bit (Block 1 (002000-003FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR2 = OFF // Table Read Protection bit (Block 2 (004000-005FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR3 = OFF // Table Read Protection bit (Block 3 (006000-007FFFh) not protected from table reads executed in other blocks)

// CONFIG7H

#pragma config EBTRB = OFF // Boot Block Table Read Protection bit (Boot block (000000-0007FFh) not protected from table reads executed in other blocks)

#define \_XTAL\_FREQ 20000000

void my\_delay(int mtime)

{

for(int i = 0; i<= mtime ; i++)

\_\_delay\_ms(1);

}

void main()

{

TRISCbits.TRISC2 = 0 ; // Set PORTC, 2 as output

TRISCbits.TRISC6 = 0 ;

TRISCbits.TRISC7 = 0 ;

PR2 = 0XFF; // set PWM period to Maximum value

CCPR1L = 0x12; // Initalise PWM duty cycle to 00

CCP1CON = 0x3C; // Configure CCP1CON as explained above.

T2CON = 0x07;

PORTCbits.RC6 = 1 ;

PORTCbits.RC7 = 0 ;

while(1)

{

CCPR1L = 0x00;// 0% duty cycle PR2 \* Duty cycle Zero SPEED

my\_delay(1500); // wait for 1.5 sec

CCPR1L = 0x33; // 20% duty cycle

my\_delay(1500);

CCPR1L = 0x66; // 40% duty cycle

my\_delay(1500);

CCPR1L = 0x99; // 60% dyty cycle

my\_delay(1500);

CCPR1L = 0xCC; // 80% dyty cycle

my\_delay(1500);

CCPR1L = 0xFF; // 100% dyty cycle Full Speed

my\_delay(1500);

}

}

**OUTPUT**

