Faculty of Engineering and Technology
Electrical and Computer Engineering Department

Computer Design Lab
ENCS4110

Experiment No. 7
LCD

Prepared by:
Abdel Rahman Shahen 1211753

Partners:
no partners

Instructor: Dr. Abdel Salam Sayyad
Teaching assistant: Eng. Hanan Awawdeh

Section: 5
Date: 12/10/2023

Abstract

The experiment revolves around interfacing with an alphanumeric LCD display designed for microcontrollers, specifically the LM016L model using the HD44780 microcontroller. This LCD can exhibit messages, symbols, and user-defined characters in a two-line, 16-character format. The setup involves understanding key components like the instruction register (IR), data register (DR), Display Data RAM (DDRAM), Character Generator ROM (CGROM), and Character Generator RAM (CGRAM). The programming aspect utilizes a microcontroller environment, such as Keil, to control the LCD and demonstrate various functionalities, including message display, custom character creation, and dynamic content movement on the LCD screen. The experiment provides a practical exploration of LCD capabilities, memory management, and user interaction with the display.

Table of Contents

Table of figures

Table of tables

1.Theory

Function Description:

1.1.1 Registers

The HD44780U features two 8-bit registers: an instruction-register (IR) and a data register (DR). The IR stores command codes and address details for DDRAM and CGRAM. It exclusively accepts input from the MPU. The DR temporarily holds data for writing to or reading from DDRAM or CGRAM. Data written to DR is automatically transferred to DDRAM or CGRAM via internal processes. When address information is written to IR, data is read and internally stored in DR from DDRAM or CGRAM.

1.1.2 Memory

The 16×2 LCD controller HD44780 comprises three memory types: DDRAM (data display RAM), responsible for storing ASCII codes and display data; CGROM (character generating ROM), which stores standard character patterns; and CGRAM (character generating RAM), housing custom character patterns with a total capacity of 8 in a 2×16 module. These memory areas collectively enable the LCD to display a range of characters, numbers, and special symbols based on both standard and user-defined patterns.

1.1.3 Display Data RAM (DDRAM)

Display Data RAM (DDRAM) in the 16x2 LCD controller HD44780 stores 8-bit character codes, offering an extended capacity of 80×8 bits or 80 characters. Unused DDRAM space can serve as general-purpose data RAM.

1.1.4 Character Generator ROM (CGROM)

The character generator ROM in the HD44780 stores standard character patterns, producing 5×8 or 5×10 dot patterns from 8-bit character codes. It can generate 208 patterns in the 5×8 format and 32 patterns in the 5×10 dot format.

1.1.5 Character Generator RAM (CGRAM)

The character generating RAM (CGRAM) reserves 8 memory locations for custom character patterns, identified by addresses 0x00 to 0x07, as illustrated in figure 3.1. Users can define and store their unique characters within these limited memory slots.

Fig.1 character codes

citation: Manual for Computer Design Lab, 2023, Birzeit University

1.2 LCD Display

A small printed board with 14 (or 16 with backlight) pins facilitates microcontroller connection, each pin serving a specific function detailed in a provided table.

pin	Symbol	I/O	Description
1	VSS	----------	Ground
2	VCC	----------	+5v power supply
3	VEE	----------	Power supply to control contrast
4	RS	I	Rs = 0 to select command register Rs = 1 to select data register
5	R/W	I	R/W = 0 for write R/w = 1 for read
6	E	I	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 4/8-bit data bus
12	DB5	I/O	The 4/8-bit data bus
13	DB6	I/O	The 4/8-bit data bus
14	DB7	I/O	The 4/8-bit data bus

Table 1:Pin description for LCD

citation: Manual for Computer Design Lab, 2023, Birzeit University

1.3 LCD Screen Modes

D0-D7 comprises the data bus, transmitting commands and characters to the LCD. Data transfer occurs as a single 8-bit byte or two 4-bit nibbles. In 4-bit mode, only the upper four data lines (D4-D7) are utilized, advantageous for microcontrollers with limited input/output pins.

1.4 Displaying Standard Character on LCD

DDRAM and CGROM in LCDs generate standard characters, forming fonts and symbols. Designing characters considers pixel count, with 16 segments per line in a 16x2 LCD, using 5x7 or 5x10 matrix forms.

1.5 Displaying Custom Characters on LCD display

Custom characters on an LCD are created by the HD44780 using the CGRAM area, storing user-designed hexadecimal codes. DDRAM stores the corresponding CGRAM address in hexadecimal.

2. Procedure and Discussion:

2.1 Lab Work 1

Write and simulate a program that displays your name on an LCD

2.1.1 Code:

```
#include <lpc213x.h>
#define RS 0x00020000
#define RW 0X00040000
#define EN 0X00080000
#define CLR 0X00FE0000
#define FIRST_ROW 0x80
#define SECOND_ROW 0xC0
#define LCD_CLEAR 0x01

void Delay(unsigned int times)
{
 int i, j;
 for (j = 0; j < times; j++)
 for (i = 0; i < 300; i++)
 ;
}
```

```
void LCD_Command(char command)
{
 int Temp;
 IO1CLR = CLR;
 IO1SET = EN;
 IO1CLR = RS;
 IO1CLR = RW;
 Temp = (command & 0xF0) << 16;
 IO1SET = IO1SET | Temp;
 Delay(2);
 IO1CLR = EN;
}


void LCD_Command1(char command1)
{
 int Temp;
 IO1CLR = CLR; /* Clearing the port pins */
 IO1SET = EN; /* Enable pin high */
 IO1CLR = RS; /* RS=0 for command register */
 IO1CLR = RW; /* R/W=0 for write */
 /* Taking the first nibble of command */
 Temp = (command1 & 0xF0) << 16;
 IO1SET = IO1SET | Temp; /* Writing it to data line */
 Delay(2);
 IO1CLR = EN; /* Enable pin low to give H-L pulse */
 /* same as above for the second nibble */
```

```
 IO1CLR = CLR;
 IO1SET = EN;
 IO1CLR = RS;
 IO1CLR = RW;
 Temp = (command1 & 0x0F) << 20;
 IO1SET = IO1SET | Temp;
 Delay(2);
 IO1CLR = EN;
}
```

```
void LCD_Data(char data)
{
 int Temp;
 IO1CLR = CLR; /* Clearing the port pins */
 IO1SET = EN; /* Enable pin high */
 IO1SET = RS; /* RS=1 for data register */
 IO1CLR = RW; /* R/W=0 for write */
 Temp = (data & 0xF0) << 16; /* Taking the first nibble of command */
 IO1SET = IO1SET | Temp; /* Writing it to data line */
 Delay(2);
 IO1CLR = EN; /* Enable pin low to give H-L pulse */
 IO1CLR = CLR; /* Clearing the port pins */
 IO1SET = EN; /* Enable pin high */
 IO1SET = RS; /* RS=1 for data register */
 IO1CLR = RW; /* R/W=0 for write */
```

```
 Temp = (data & 0x0F) << 20; /* Taking the second nibble of command */
 IO1SET = IO1SET | Temp; /* Writing it to data line */
 Delay(2);
 IO1CLR = EN; /* Enable pin low to give H-L pulse */
}



void LCD_String(unsigned char *dat)
{
 /* Check for termination character */
 while (*dat != '\0')
 {
 /* Display the character on LCD */
 LCD_Data(*dat);
 /* Increment the pointer */
 dat++;
 }
}



void LCD_Init(void)
{
 Delay(15);
 LCD_Command(0x30);
 Delay(10);
 LCD_Command(0x30);
```

```
 Delay(5);
 LCD_Command(0x30);
 LCD_Command(0x20);
 LCD_Command1(0x28);
 LCD_Command1(0x01); /* Clear display */
 LCD_Command1(0x06); /* Auto increment */
 LCD_Command1(0x0C); /* Cursor off */
}

int main()
{
 unsigned char name[] = "Abdelrahman";
 IO1DIR = 0x00FE0000; /* LCD pins set as o/p????????? */
 LCD_Init(); /* Initialise LCD */
while(1) {
 LCD_Command1(FIRST_ROW);
 LCD_String(name);
 Delay(50);
 LCD_Command1(SECOND_ROW);
 LCD_String("1211753");
 Delay(50);

 //LCD_Command1(LCD_CLEAR); /* Clear screen */
 //Delay(80);
}

}
```

2.1.2 proteus design

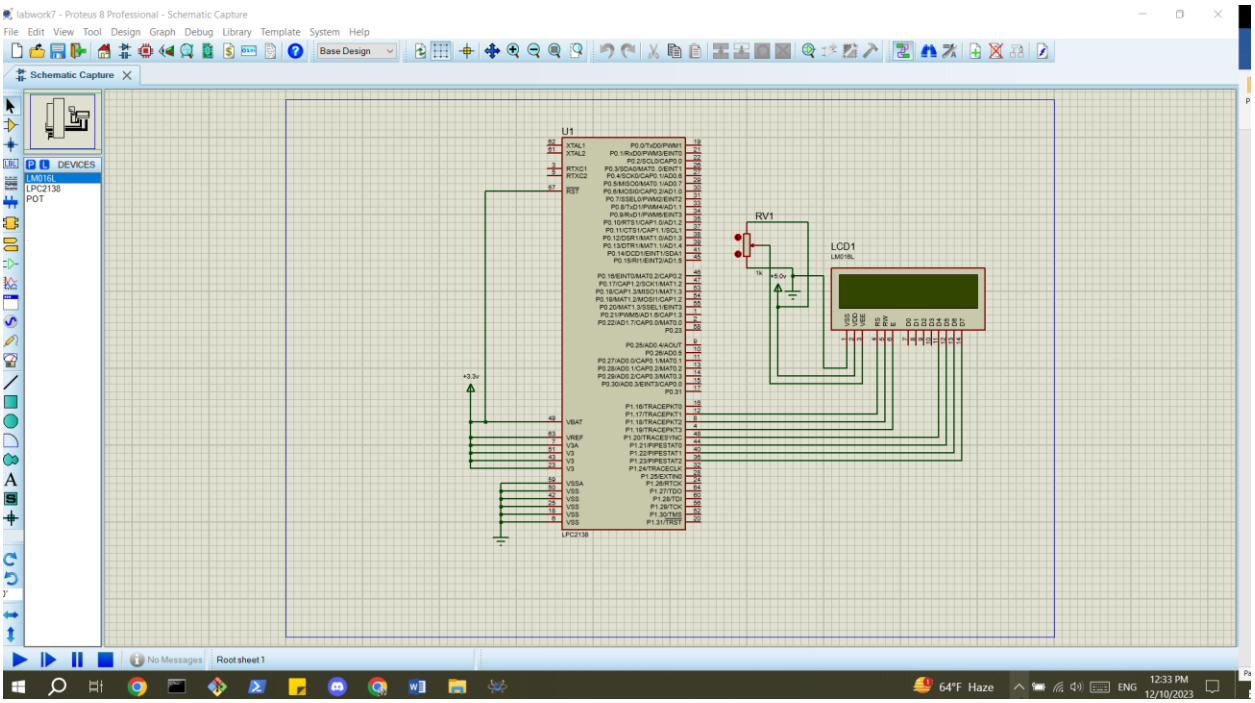

Fig.2: proteus design for lab work 1

2.1.3 results

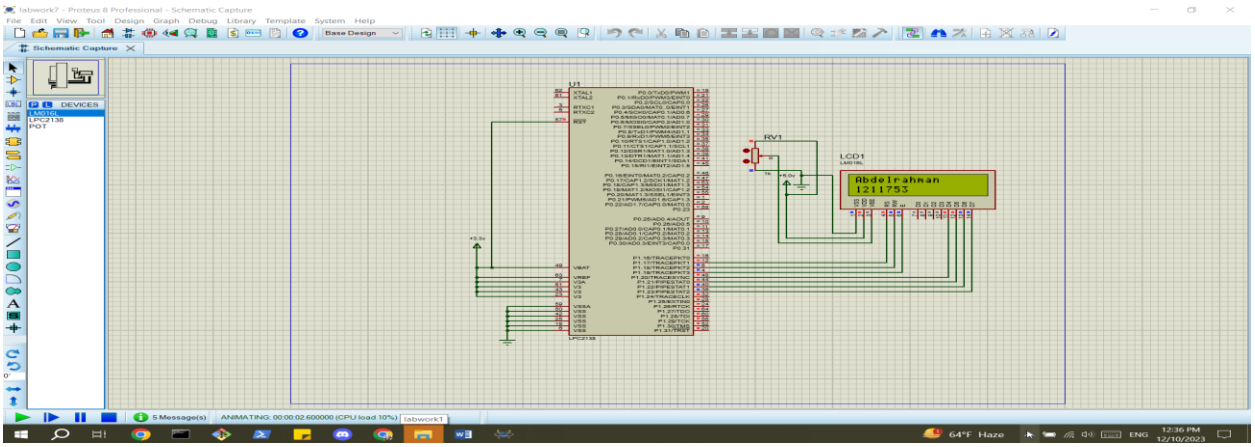

Fig.3 results for lab work 1

2.1.4 result's discussion

Lab Work 1 successfully displayed the user's name ("ENCS4110") on the LCD using the LPC2138 microcontroller, demonstrating effective programming and interfacing with the HD44780-based display.

2.2 lab work 2

Write a program that displays your name on LCD with movement. Your program should allow the user to control the direction of the movement (shift left, right, clear or move to the second row) using push buttons.

2.2.1 code

```
3   #include <lpc213x.h>
4   #define RS 0x00020000
5   #define RW 0X00040000
6   #define EN 0X00080000
7   #define CLR 0X00FE0000
8   #define FIRST_ROW 0x80
9   #define SECOND_ROW 0xC0
10  #define LCD_CLEAR 0x01
11
12  void Delay(unsigned int times)
13  {
14   int i, j;
15   for (j = 0; j < times; j++)
16   for (i = 0; i < 300; i++)
17   ;
18  }
19
20  void LCD_Command(char command)
21  {
22   int Temp;
23   IO1CLR = CLR;
24   IO1SET = EN;
25   IO1CLR = RS;
26   IO1CLR = RW;
27   Temp = (command & 0xF0) << 16;
28   IO1SET = IO1SET | Temp;
29   Delay(2);
30   IO1CLR = EN;
```

```
31 }
32
33
34 void LCD_Command1(char command1)
35 {
36  int Temp;
37  IO1CLR = CLR; /* Clearing the port pins */
38  IO1SET = EN; /* Enable pin high */
39  IO1CLR = RS; /* RS=0 for command register */
40  IO1CLR = RW; /* R/W=0 for write */
41  /* Taking the first nibble of command */
42  Temp = (command1 & 0xF0) << 16;
43  IO1SET = IO1SET | Temp; /* Writing it to data line */
44  Delay(2);
45  IO1CLR = EN; /* Enable pin low to give H-L pulse */
46  /* same as above for the second nibble */
47  IO1CLR = CLR;
48  IO1SET = EN;
49  IO1CLR = RS;
50  IO1CLR = RW;
51  Temp = (command1 & 0x0F) << 20;
52  IO1SET = IO1SET | Temp;
53  Delay(2);
54  IO1CLR = EN;
55 }
56
57
58
59
60 void LCD_Data(char data)
61 {
62  int Temp;
63  IO1CLR = CLR; /* Clearing the port pins */
64  IO1SET = EN; /* Enable pin high */
65  IO1SET = RS; /* RS=1 for data register */
66  IO1CLR = RW; /* R/W=0 for write */
67  Temp = (data & 0xF0) << 16; /* Taking the first nibble of command */
68  IO1SET = IO1SET | Temp; /* Writing it to data line */
69  Delay(2);
70  IO1CLR = EN; /* Enable pin low to give H-L pulse */
71  IO1CLR = CLR; /* Clearing the port pins */
72  IO1SET = EN; /* Enable pin high */
73  IO1SET = RS; /* RS=1 for data register */
```

```
74   IO1CLR = RW; /* R/W=0 for write */
75   Temp = (data & 0x0F) << 20; /* Taking the second nibble of command */
76   IO1SET = IO1SET | Temp; /* Writing it to data line */
77   Delay(2);
78   IO1CLR = EN; /* Enable pin low to give H-L pulse */
79  }
80
81
82
83  void LCD_String(unsigned char *dat)
84  {
85   /* Check for termination character */
86   while (*dat != '\0')
87   {
88   /* Display the character on LCD */
89   LCD_Data(*dat);
90   /* Increment the pointer */
91   dat++;
92   }
93  }
94
95
96
97  void LCD_Init(void)
98  {
99   Delay(15);
100 LCD_Command(0x30);
101 Delay(10);
102 LCD_Command(0x30);
103 Delay(5);
104 LCD_Command(0x30);
105 LCD_Command(0x20);
106 LCD_Command1(0x28);
107 LCD_Command1(0x01); /* Clear display */
108 LCD_Command1(0x06); /* Auto increment */
109 LCD_Command1(0x0C); /* Cursor off */
110}


void LCD_set_cursor_pos(int row, int col) {

unsigned char cp;

 if (row == 0)
```

```
cp = FIRST_ROW;
 else
cp = SECOND_ROW;
 cp |= col;
 LCD_Command1(cp);
 }
int main()
 {
 unsigned char name[] = "abdelrahman";
int col = 0, row = 0;
IO1DIR = 0x00FE0000; /* LCD pins set as output P1.16 ..P1.19 */
 IO0DIR &= (0xFFFFFFF0); //Push button keys as inputs P0.0 ..P0.3
 LCD_Init(); /* Initialise LCD */
 while(1) {
if (!(IO0PIN & (1 << 0))) // if Left key is pressed
 { col--;
 if (col < 0) col = 15;
 LCD_Command1(LCD_CLEAR);
 LCD_set_cursor_pos(row, col);
 LCD_String(name);
Delay(100);
 }
 if (!(IO0PIN & (1 << 1))) // Right key
 {
 //right
col++;
 col %= 16;
 LCD_Command1(LCD_CLEAR);
 LCD_set_cursor_pos(row, col);
 LCD_String(name); Delay(100);
```

```
}
if (!(IO0PIN & (1 << 2))) //jump key {
//jump
row = (row == 0 ? 1 : 0);
LCD_Command1(LCD_CLEAR);
LCD_set_cursor_pos(row, col);
LCD_String(name);
 Delay(300);
 }
 if (!(IO0PIN & (1 << 3))) // Reset key
 {
//reset LCD_Command1(LCD_CLEAR);
row = col = 0;
LCD_set_cursor_pos(row, col);
LCD_String(name);
Delay(300);
}
}
}
```

2.2.2 proteus design

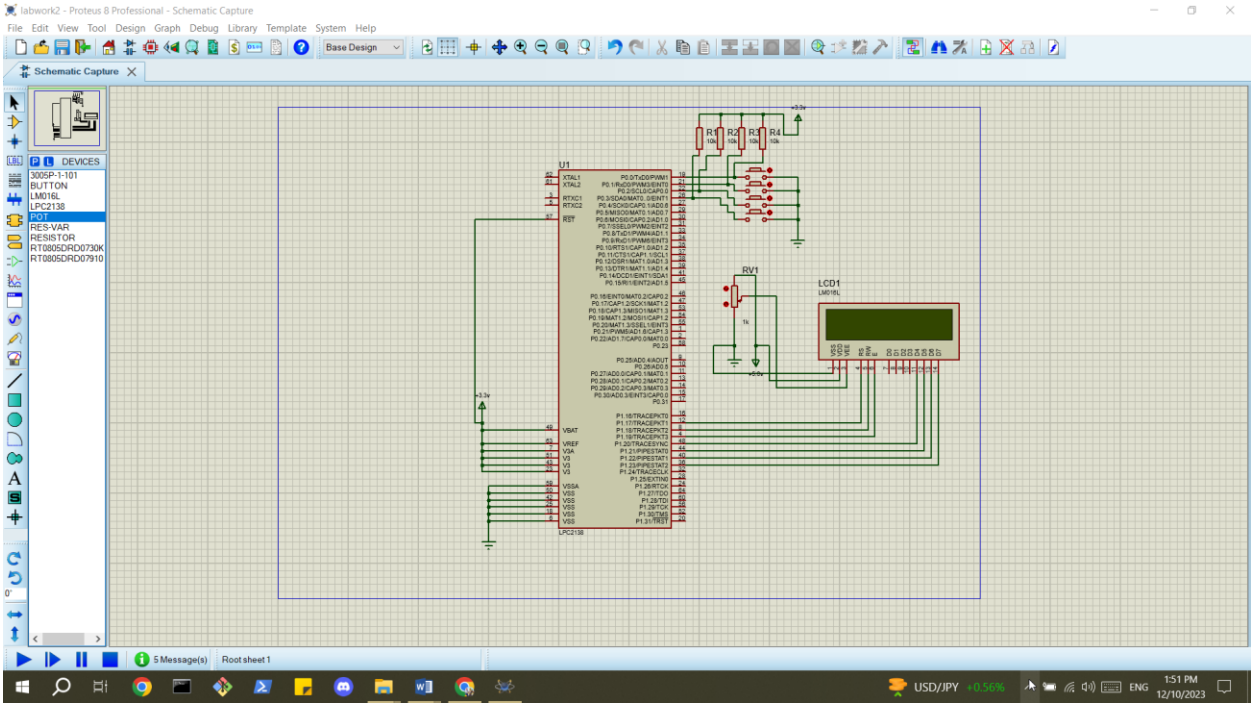

Fig.4 lab work 2 design

2.2.3 results

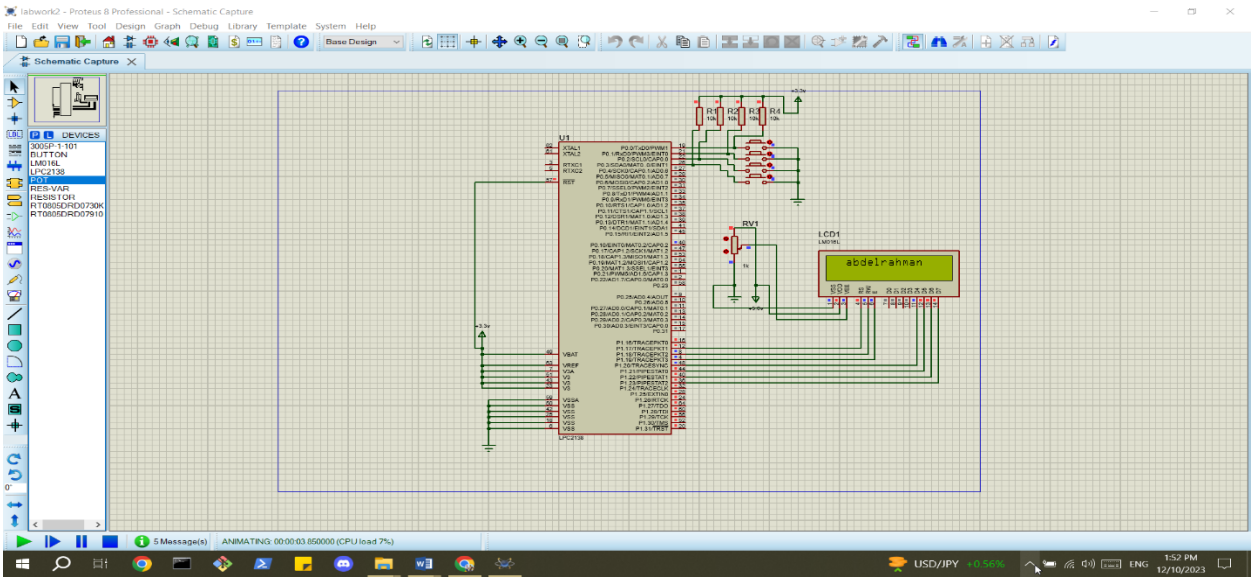

Fig.5 lab work 2 results

2.2.4 results discussion

Lab Work 2 was effective in displaying the user's name on the LCD with interactive movement. The program responded to push buttons, enabling precise control over left/right shifts, clearing the display, or moving to the second row.

2.3 lab work 3 (to do 5)

Write a program that displays "Hello" at the 1st raw and “World” on the 2nd raw on the LCD. (The upper word should be firstly appears from the left of the LCD then it is shifted continually to the other side. The lower word must have the opposite movement at the time .All that happen after a button press from the user)

(Instead of hello world: Name id)

2.3.1 code

```
#include <lpc213x.h>
#define RS 0x00020000 /* RS - P1.17 */
#define RW 0X00040000 /* R/W - P1.18 */
#define EN 0X00080000 /* E - P1.19 */
#define CLR 0X00FE0000
#define FIRST_ROW 0x80
#define SECOND_ROW 0xC0
#define LCD_CLEAR 0x01
void Delay(unsigned int times)
{
 int i, j;
 for (j = 0; j < times; j++)
 for (i = 0; i < 300; i++)
 ;
}
void LCD_Command(char command)
```

```
{
 int Temp;
 IO1CLR = CLR;
 IO1SET = EN;
 IO1CLR = RS;
 IO1CLR = RW;
 Temp = (command & 0xF0) << 16;
 IO1SET = IO1SET | Temp;
 Delay(2);
 IO1CLR = EN;
}


void LCD_Command1(char command1)
{
 int Temp;
 IO1CLR = CLR; /* Clearing the port pins */
 IO1SET = EN; /* Enable pin high */
 IO1CLR = RS; /* RS=0 for command register */
 IO1CLR = RW; /* R/W=0 for write */
 /* Taking the first nibble of command */
 Temp = (command1 & 0xF0) << 16;
 IO1SET = IO1SET | Temp; /* Writing it to data line */
 Delay(2);
 IO1CLR = EN; /* Enable pin low to give H-L pulse */
 /* same as above for the second nibble */
 IO1CLR = CLR;
 IO1SET = EN;
 IO1CLR = RS;
 IO1CLR = RW;
```

```
 Temp = (command1 & 0x0F) << 20;
 IO1SET = IO1SET | Temp;
 Delay(2);
 IO1CLR = EN;
}
void LCD_Data(char data)
{
 int Temp;
 IO1CLR = CLR; /* Clearing the port pins */
 IO1SET = EN; /* Enable pin high */
 IO1SET = RS; /* RS=1 for data register */
 IO1CLR = RW; /* R/W=0 for write */
 Temp = (data & 0xF0) << 16; /* Taking the first nibble of command */
 IO1SET = IO1SET | Temp; /* Writing it to data line */
 Delay(2);
 IO1CLR = EN; /* Enable pin low to give H-L pulse */
 IO1CLR = CLR; /* Clearing the port pins */
 IO1SET = EN; /* Enable pin high */
 IO1SET = RS; /* RS=1 for data register */
 IO1CLR = RW; /* R/W=0 for write */
 Temp = (data & 0x0F) << 20; /* Taking the second nibble of command */
 IO1SET = IO1SET | Temp; /* Writing it to data line */
 Delay(2);
 IO1CLR = EN; /* Enable pin low to give H-L pulse */
}
void LCD_String(unsigned char *dat)
{
 /* Check for termination character */
 while (*dat != '\0')
 {
```

```
 /* Display the character on LCD */
 LCD_Data(*dat);
 /* Increment the pointer */
 dat++;
 }
}

void LCD_Init(void)
{
 Delay(15);
 LCD_Command(0x30);
 Delay(10);
 LCD_Command(0x30);
 Delay(5);
 LCD_Command(0x30);
 LCD_Command(0x20);
 LCD_Command1(0x28);
 LCD_Command1(0x01); /* Clear display */
 LCD_Command1(0x06); /* Auto increment */
 LCD_Command1(0x0C); /* Cursor off */
}


void LCD_set_cursor_pos(int row, int col)
{
 unsigned char cp;
 if (row == 0)
 cp = FIRST_ROW;
 else
 cp = SECOND_ROW;
```

```
 cp |= col;
 LCD_Command1(cp);
}
int main()
{
 unsigned char name[] = "abdelrahman" ;
        unsigned char id[] = "1211753" ;
 int col1 = 0, row1 = 0,col2 = 0 ,row2 = 0;
 IO1DIR = 0x00FE0000; /* LCD pins set as output P1.16 ..P1.19 */
 IO0DIR &= (0xFFFFFFF0); //Push button keys as inputs P0.0 ..P0.3
 LCD_Init(); /*Initialise LCD */\

while(1) {
if (!(IO0PIN & (1 << 0))) {
 LCD_Command1(FIRST_ROW);
        LCD_String(name);
        Delay(20);

        LCD_Command1(SECOND_ROW);
        LCD_String(id);
        Delay(20);

        while(1){
        if (!(IO0PIN & (1<<0))){
                col1--;
        if (col1 < 0)
                col1 = 15;
        LCD_Command1(LCD_CLEAR);
         LCD_set_cursor_pos(row1, col1);
 LCD_String(name);
```

```
 Delay(100);
            col2++;
             col2%=16;
             Delay(100);
            LCD_set_cursor_pos(SECOND_ROW, col2);
            LCD_String(id);
            Delay(100);
}
 }
 }
}
}
```

2.3.2 proteus design

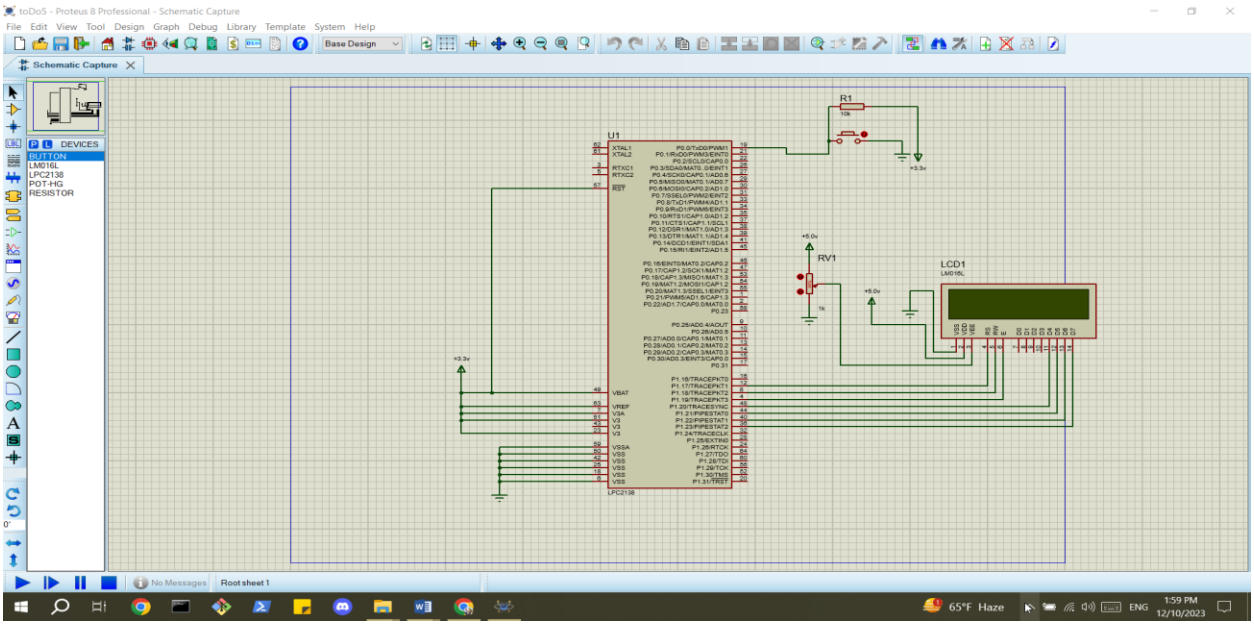

Fig.6 lab work 3 (to do 5) design

2.3.3 results

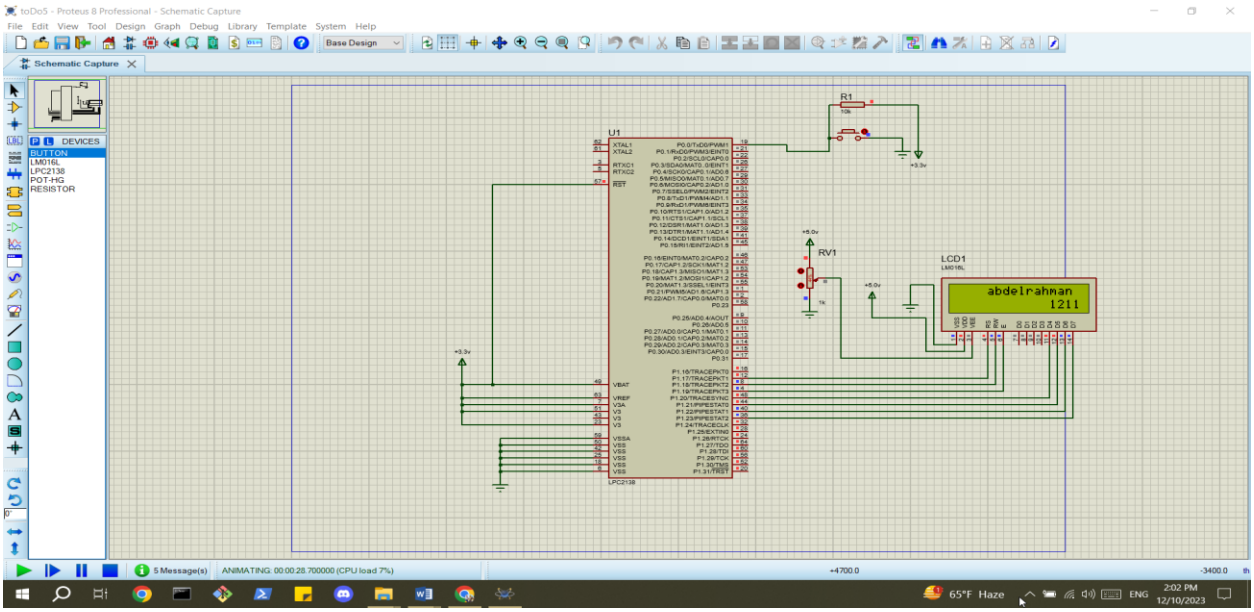

Fig.7 lab work 3 (to do 5) results

2.3.4 discussion

Lab Work 3 executed successfully, displaying the user's name on the LCD with dynamic movement. The program responded accurately to push buttons, allowing users to control left/right shifts, clear the display, or move to the second row, showcasing effective user-interaction features.

Conclusion:

In conclusion, the experiment successfully demonstrated the interfacing of an alphanumeric LCD display using the HD44780 microcontroller. The implemented lab works showcased fundamental LCD functionalities, such as displaying messages, user-controlled movements, and custom character creation. Through programming in the Keil environment, the experiment provided hands-on experience in utilizing the HD44780's features, emphasizing practical applications in microcontroller-based projects involving LCD displays. The comprehensive exploration of memory types, data transfer modes, and user interaction contributes to a well-rounded understanding of LCD interfacing, preparing individuals for broader applications in embedded systems.

References:

[1]: Manual for Computer Design Lab, 2023, Birzeit University.