

**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**

[**Abstract** I](#_Toc153109956)

[**Table of Contents** II](#_Toc153109957)

[**Table of figures** III](#_Toc153109958)

[**Table of tables** IV](#_Toc153109959)

[**1.Theory** 1](#_Toc153109960)

[**Function Description:** 1](#_Toc153109961)

[**1.1.1 Registers** 1](#_Toc153109962)

[**1.1.2 Memory** 1](#_Toc153109963)

[**1.1.3 Display Data RAM (DDRAM)** 1](#_Toc153109964)

[**1.1.4 Character Generator ROM (CGROM)** 1](#_Toc153109965)

[**1.2** **LCD Display** 3](#_Toc153109966)

[**1.3** **LCD Screen Modes** 3](#_Toc153109967)

[**1.4** **Displaying Standard Character on LCD** 3](#_Toc153109968)

[**1.5** **Displaying Custom Characters on LCD display** 4](#_Toc153109969)

[**2. Procedure and Discussion:** 4](#_Toc153109970)

[**2.1** **Lab Work 1** 4](#_Toc153109971)

[**2.1.1** **Code:** 4](#_Toc153109972)

[**2.1.2** **proteus design** 9](#_Toc153109973)

[**2.1.3** **results** 9](#_Toc153109974)

[**2.1.4** **result’s discussion** 11](#_Toc153109975)

[**2.2** **lab work 2** 11](#_Toc153109976)

[**2.2.1** **code** 11](#_Toc153109977)

[**2.2.2 proteus design** 16](#_Toc153109978)

[**2.2.3 results** 16](#_Toc153109979)

[2.2.4 results discussion 17](#_Toc153109980)

[**2.3 lab work 3 (to do 5)** 17](#_Toc153109981)

[**2.3.1 code** 17](#_Toc153109982)

[**2.3.2 proteus design** 23](#_Toc153109983)

[**2.3.3 results** 23](#_Toc153109984)

[**2.3.4 discussion** 24](#_Toc153109985)

[**Conclusion:** 24](#_Toc153109986)

[**References** 24](#_Toc153109987)

# **Table of figures**

Fig.1: *character codes ...…..………………………………………………………………………….....2*

[Fig.2: proteus design for lab work 1 9](#_Toc153109774)

[Fig.3 results for lab work 1 9](#_Toc153109775)

[Fig.4 lab work 2 design 15](#_Toc153109776)

[Fig.5 lab work 2 results 15](#_Toc153109777)

[Fig.6 lab work 3 (to do 5) design 22](#_Toc153109778)

[Fig.7 lab work 3 (to do 5) results 22](#_Toc153109779)

# **Table of tables**

[Table 1:Pin description for LCD 3](#_Toc153109719)

# **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

## **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

## **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.

## **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.

## **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:**

## **Lab Work 1**

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

### **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);

}

}

### **proteus design**

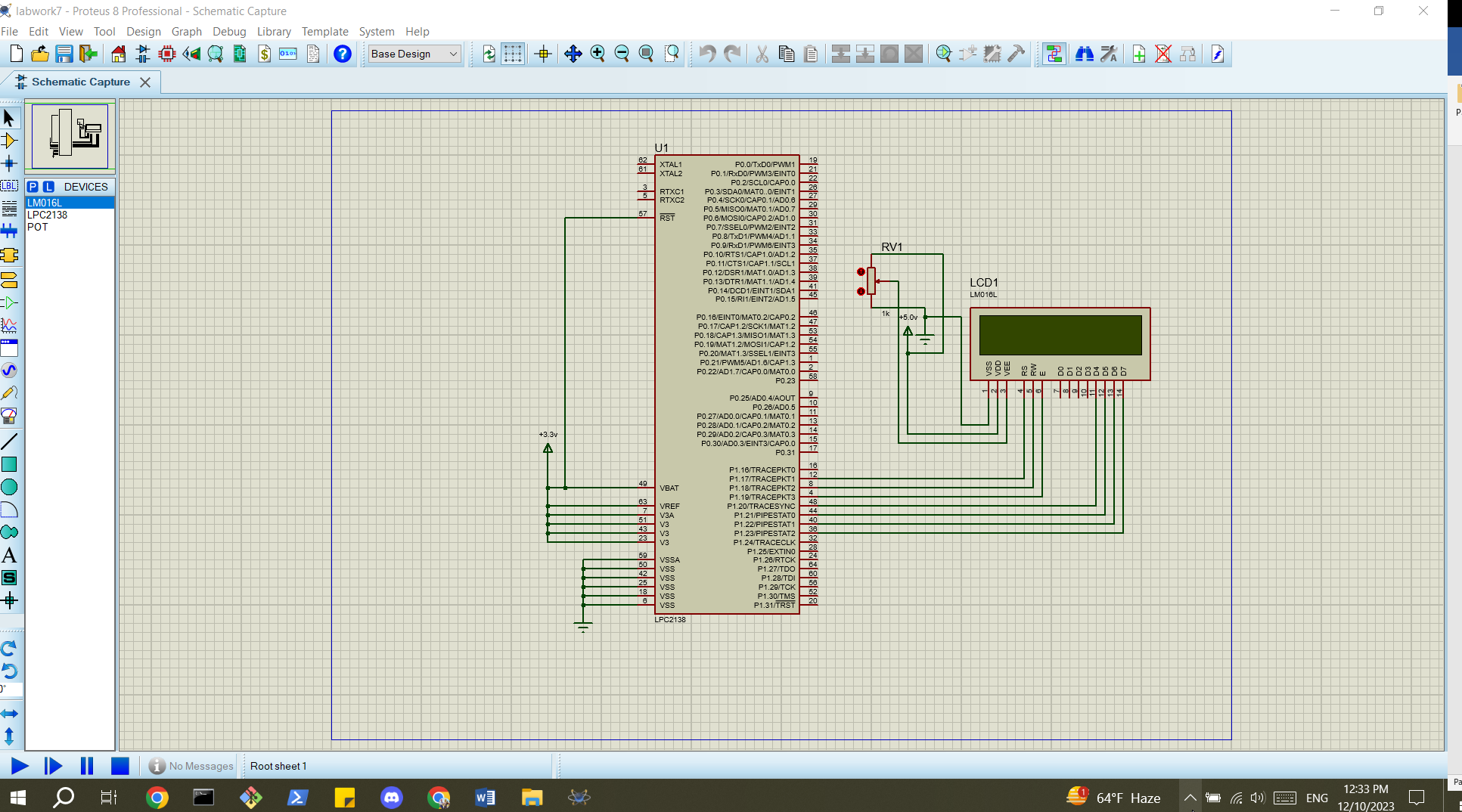


Fig.2: proteus design for lab work 1

### **results**

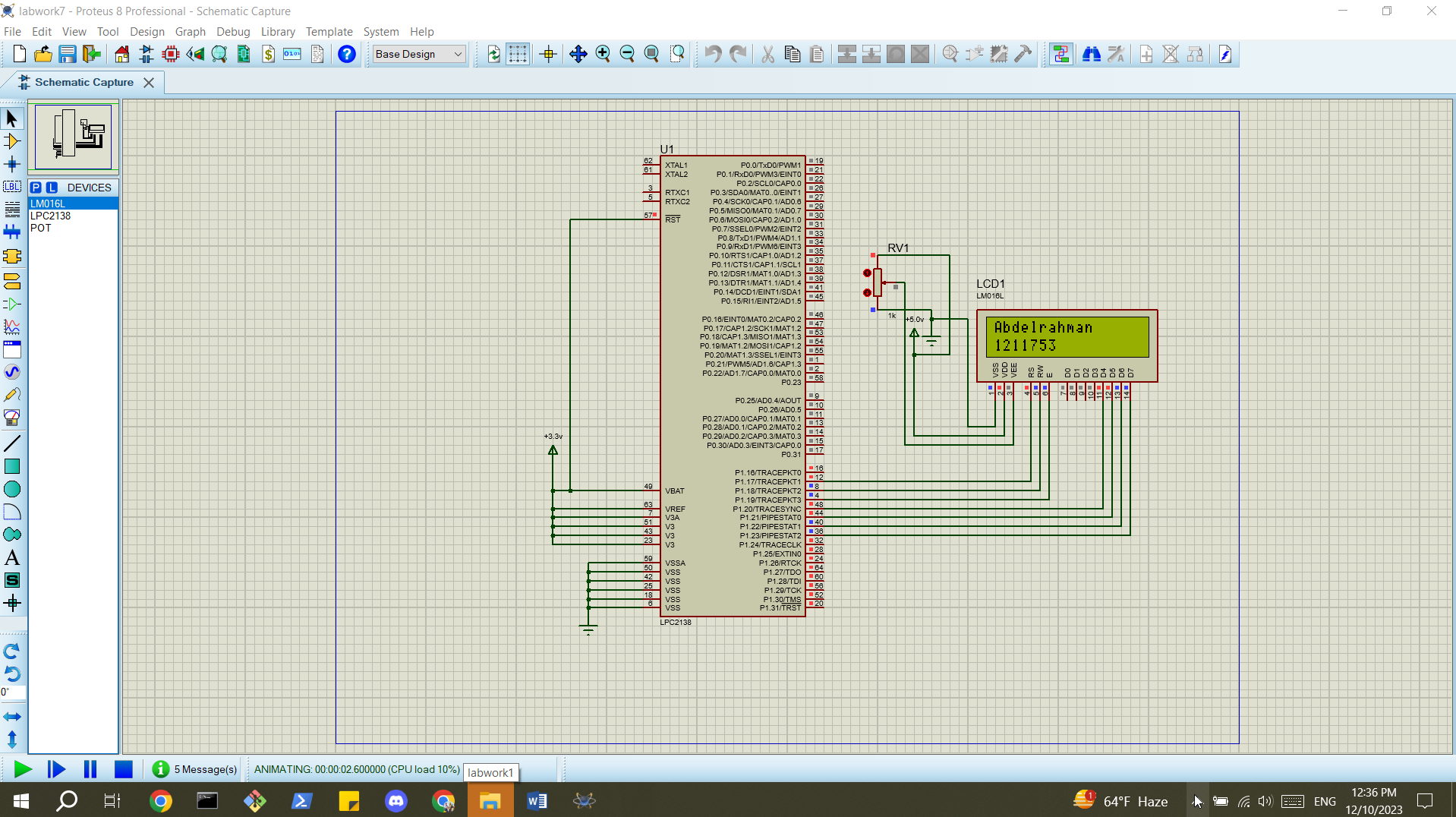


Fig.3 results for lab work 1

### **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.

## **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.

### **code**

1. #include <lpc213x.h>
2. #define RS 0x00020000
3. #define RW 0X00040000
4. #define EN 0X00080000
5. #define CLR 0X00FE0000
6. #define FIRST\_ROW 0x80
7. #define SECOND\_ROW 0xC0
8. #define LCD\_CLEAR 0x01
9. void Delay(unsigned int times)
10. {
11. int i, j;
12. for (j = 0; j < times; j++)
13. for (i = 0; i < 300; i++)
14. ;
15. }
16. void LCD\_Command(char command)
17. {
18. int Temp;
19. IO1CLR = CLR;
20. IO1SET = EN;
21. IO1CLR = RS;
22. IO1CLR = RW;
23. Temp = (command & 0xF0) << 16;
24. IO1SET = IO1SET | Temp;
25. Delay(2);
26. IO1CLR = EN;
27. }
28. void LCD\_Command1(char command1)
29. {
30. int Temp;
31. IO1CLR = CLR; /\* Clearing the port pins \*/
32. IO1SET = EN; /\* Enable pin high \*/
33. IO1CLR = RS; /\* RS=0 for command register \*/
34. IO1CLR = RW; /\* R/W=0 for write \*/
35. /\* Taking the first nibble of command \*/
36. Temp = (command1 & 0xF0) << 16;
37. IO1SET = IO1SET | Temp; /\* Writing it to data line \*/
38. Delay(2);
39. IO1CLR = EN; /\* Enable pin low to give H-L pulse \*/
40. /\* same as above for the second nibble \*/
41. IO1CLR = CLR;
42. IO1SET = EN;
43. IO1CLR = RS;
44. IO1CLR = RW;
45. Temp = (command1 & 0x0F) << 20;
46. IO1SET = IO1SET | Temp;
47. Delay(2);
48. IO1CLR = EN;
49. }
50. void LCD\_Data(char data)
51. {
52. int Temp;
53. IO1CLR = CLR; /\* Clearing the port pins \*/
54. IO1SET = EN; /\* Enable pin high \*/
55. IO1SET = RS; /\* RS=1 for data register \*/
56. IO1CLR = RW; /\* R/W=0 for write \*/
57. Temp = (data & 0xF0) << 16; /\* Taking the first nibble of command \*/
58. IO1SET = IO1SET | Temp; /\* Writing it to data line \*/
59. Delay(2);
60. IO1CLR = EN; /\* Enable pin low to give H-L pulse \*/
61. IO1CLR = CLR; /\* Clearing the port pins \*/
62. IO1SET = EN; /\* Enable pin high \*/
63. IO1SET = RS; /\* RS=1 for data register \*/
64. IO1CLR = RW; /\* R/W=0 for write \*/
65. Temp = (data & 0x0F) << 20; /\* Taking the second nibble of command \*/
66. IO1SET = IO1SET | Temp; /\* Writing it to data line \*/
67. Delay(2);
68. IO1CLR = EN; /\* Enable pin low to give H-L pulse \*/
69. }
70. void LCD\_String(unsigned char \*dat)
71. {
72. /\* Check for termination character \*/
73. while (\*dat != '\0')
74. {
75. /\* Display the character on LCD \*/
76. LCD\_Data(\*dat);
77. /\* Increment the pointer \*/
78. dat++;
79. }
80. }
81. void LCD\_Init(void)
82. {
83. Delay(15);
84. LCD\_Command(0x30);
85. Delay(10);
86. LCD\_Command(0x30);
87. Delay(5);
88. LCD\_Command(0x30);
89. LCD\_Command(0x20);
90. LCD\_Command1(0x28);
91. LCD\_Command1(0x01); /\* Clear display \*/
92. LCD\_Command1(0x06); /\* Auto increment \*/
93. LCD\_Command1(0x0C); /\* Cursor off \*/
94. }

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 designC:\Users\mrabd\Documents\ShareX\Screenshots\2023-12\U4HmExjUcn.png**

Fig.4 lab work 2 design

### **2.2.3 results**

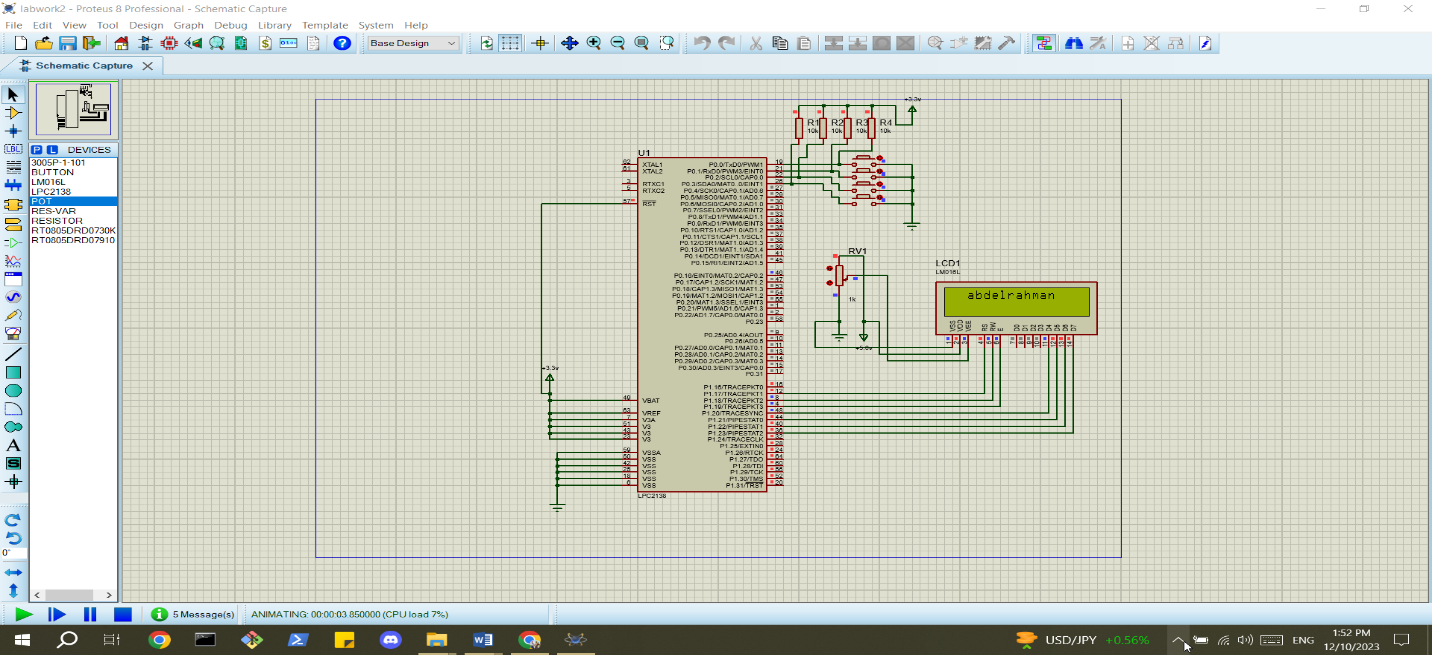


Fig.5 lab work 2 results

### 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**

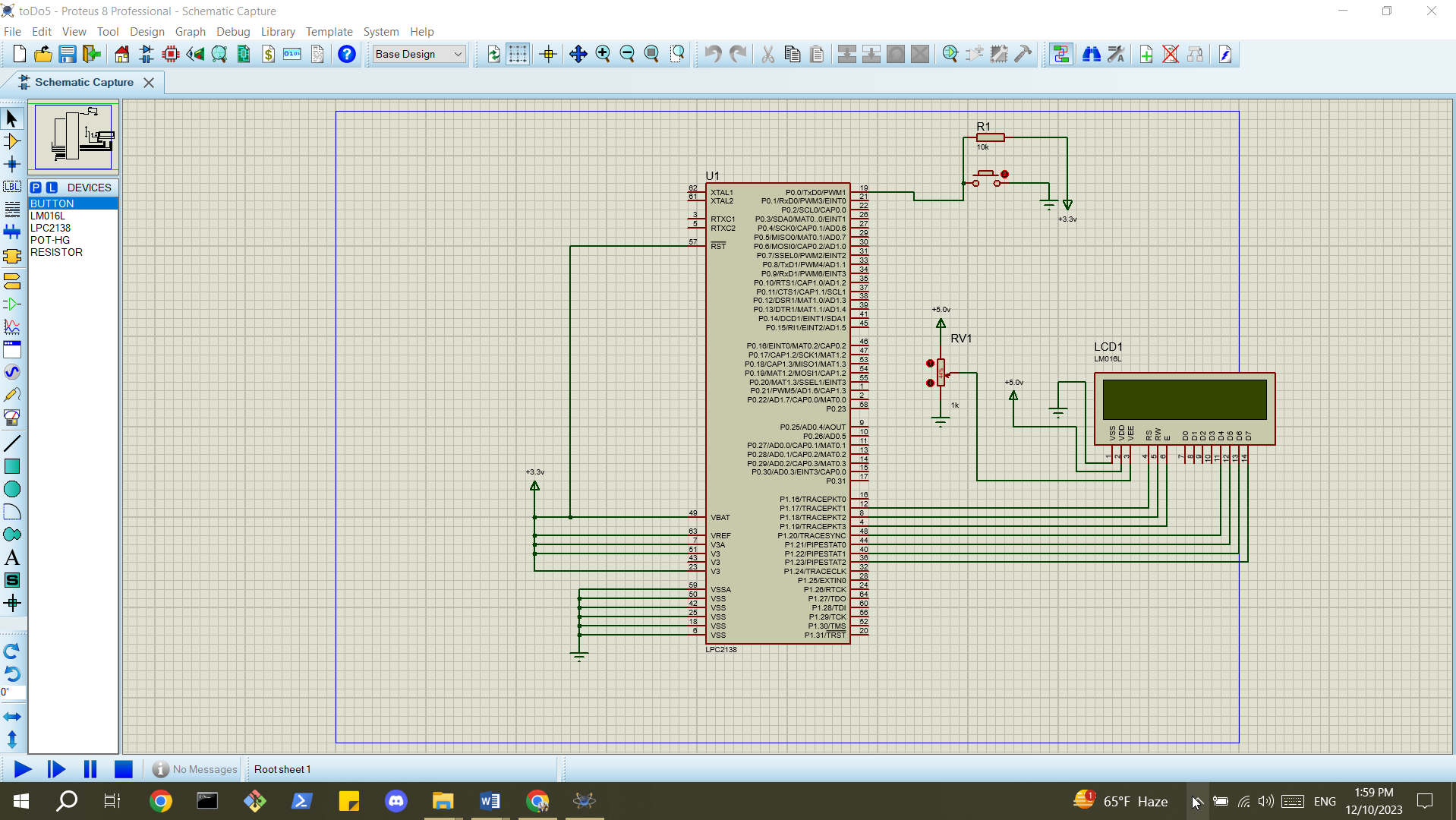


Fig.6 lab work 3 (to do 5) design

### **2.3.3 results**

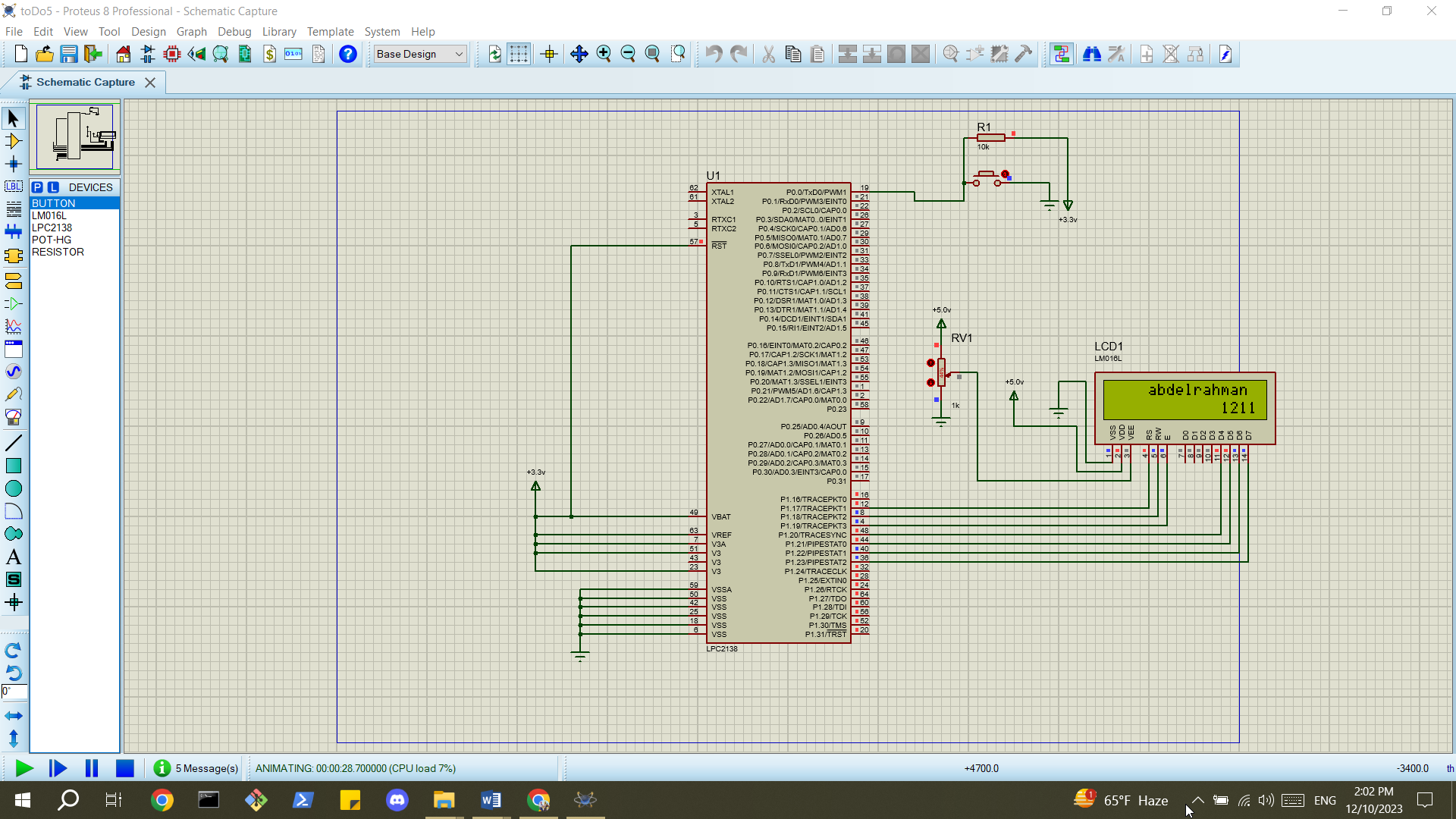


Fig.7 lab work 3 (to do 5) results

### **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.