

**Faculty of Engineering and Technology**

**Electrical and Computer Engineering Department**

**Computer Design Lab**

**ENCS4110**

**Experiment No. 10**

**Keypad**

Prepared by:

**Abdel Rahman Shahen** **1211753**

Partners:

 no partners

Instructor: Dr. Abdel Salam Sayyad

Teaching assistant: Eng. Hanan Awawdeh

Section: 5

Date: 1/13/2024

# **Abstract**

This experiment details the interfacing of a 12-key matrix keypad with an LPC2138 microcontroller. It focuses on the practical aspects of connecting and using a matrix keypad, where keys are arranged in rows and columns to minimize the required microcontroller pins. The experiment involves programming the microcontroller to detect key presses and display the corresponding key value on a 7-segment display, demonstrating the integration of input devices with microcontrollers.

Table of Contents

[**Abstract** I](#_Toc156003012)

[**Table of figures:** III](#_Toc156003013)

[**Table of tables** IV](#_Toc156003014)

[**1.** **Theory** 1](#_Toc156003015)

[**1.1 Matrix Keypad Interface with LPC2138 Microcontroller** 1](#_Toc156003016)

[**2.** **Procedure and Discussion:** 2](#_Toc156003017)

[**2.1 Lab Work 1** 2](#_Toc156003018)

[**2.1.1 code** 2](#_Toc156003019)

[**2.1.2 proteus design** 5](#_Toc156003020)

[**2.1.3 results** 5](#_Toc156003022)

[**2.1.4 result’s Discussion** 6](#_Toc156003023)

[**2.2** **To do task 1** 6](#_Toc156003024)

[**2.2.1** **code** 6](#_Toc156003025)

[**2.2.2** **proteus design** 9](#_Toc156003026)

[**2.2.3** **results** 9](#_Toc156003027)

[**2.2.4** **result’s discussion** 10](#_Toc156003028)

[**2.3** **to-do 2** 10](#_Toc156003029)

[**2.3.1 code** 10](#_Toc156003030)

[**2.3.2 proteus design** 13](#_Toc156003031)

[**2.3.3 result** 13](#_Toc156003032)

[**2.3.4 result’s discussion** 14](#_Toc156003033)

[Conclusion: 15](#_Toc156003034)

[**References** 18](#_Toc156003035)

# **Table of figures:**

[Fig.1 : Matrix Keypad Interface 1](#_Toc156002755)

[**Fig.2: proteus design for labWork1** 5](#_Toc156002756)

[**Fig.3: results for labWork1** 5](#_Toc156002756)

[**Fig.4: proteus design for to-do 1** 9](#_Toc156002756)

[**Fig.5: results for to-do 1** 9](#_Toc156002756)

[**Fig.6: proteus design for to-do 2** 13](#_Toc156002756)

**[Fig.7: results for to-do 2](#_Toc156002756)** [13](#_Toc156002756)

# **Table of tables**

Top of Form

# **Theory**

## **1.1 Matrix Keypad Interface with LPC2138 Microcontroller**

The keypad in "Exp10: Keypad" works using a matrix arrangement to efficiently interface with the LPC2138 microcontroller. The 12 keys of the keypad are organized into 3 columns and 4 rows. Each key, when pressed, connects its corresponding row to the column, forming an electrical contact. The columns are connected to three output pins, and the rows to four input pins of the LPC2138. To detect a key press, the microcontroller outputs a logic '0' to each column sequentially while scanning the row values. If a row reads logic '0', it indicates that the key at the intersection of the current column and row is pressed. This process is repeated continuously to detect key presses. This setup allows for efficient utilization of microcontroller pins, reducing the number needed for interfacing with the keypad.

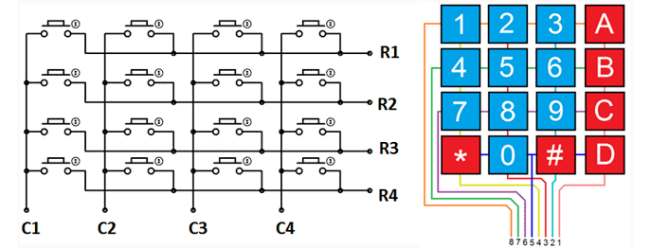


Fig.1: Matrix Keypad Interface

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

# **Procedure and Discussion:**

## **2.1 Lab Work 1**

Write and simulate a program that displays the value of the pressed key on a 7-seg.

### **2.1.1 code**

#include<lpc213x.h>

#define bit(x) 1<<x

void msDelay(int d) {

int i,j;

long c=0;

d=d\*2;

for(i=0;i<d;i++){

for(j=0;j<1000;j++){

c++;

}

}

}

int dec\_to\_7seg(int number)

{

switch(number){

case 0 : return 0x3F;

case 1 : return 0x06;

case 2 : return 0x5B;

case 3 : return 0x4F;

case 4 : return 0x66;

case 5 : return 0x6D;

case 6 : return 0x7D;

case 7 : return 0x07;

case 8 : return 0x7F;

case 9 : return 0x6F;

default : return 0x00;

}

}

int main(){

IO0DIR |= 0x000000FF; //P0.0 ...P0.7 output

IO1DIR &= ~(0x000F0000); //P1.16 ...P1.19 input

IO1DIR |= 0x00700000; // P1.20 ...P1.22 output

while(1){

//test first column

IO1CLR = bit(20);

IO1SET = bit(21)| bit(22);

if(!(IO1PIN & bit(16))){

msDelay(50);

IO0CLR = 0x000000FF;

IO0PIN |= dec\_to\_7seg(1);

}

else if (!(IO1PIN & bit(17))){

msDelay(50);

IO0CLR = 0x000000FF;

IO0PIN |= dec\_to\_7seg(4);

}

else if (!(IO1PIN & bit(18))){

msDelay(50);

IO0CLR = 0x000000FF;

IO0PIN |= dec\_to\_7seg(7);

}

//test second column

IO1CLR = bit(21);

IO1SET = bit(20)| bit(22);

if(!(IO1PIN & bit(16))){

msDelay(50);

IO0CLR = 0x000000FF;

IO0PIN |= dec\_to\_7seg(2);

}

else if (!(IO1PIN & bit(17))){

msDelay(50);

IO0CLR = 0x000000FF;

IO0PIN |= dec\_to\_7seg(5);

}

else if (!(IO1PIN & bit(18))){

msDelay(50);

IO0CLR = 0x000000FF;

IO0PIN |= dec\_to\_7seg(8);

}

else if (!(IO1PIN & bit(19))){

msDelay(50);

IO0CLR = 0x000000FF;

IO0PIN |= dec\_to\_7seg(0);

}

// test third column

IO1CLR = bit(22);

IO1SET = bit(20)| bit(21);

if(!(IO1PIN & bit(16))){

msDelay(50);

IO0CLR = 0x000000FF;

IO0PIN |= dec\_to\_7seg(3);

}

else if (!(IO1PIN & bit(17))){

msDelay(50);

IO0CLR = 0x000000FF;

IO0PIN |= dec\_to\_7seg(6);

}

else if (!(IO1PIN & bit(18))){

msDelay(50);

IO0CLR = 0x000000FF;

IO0PIN |= dec\_to\_7seg(9);

}

}

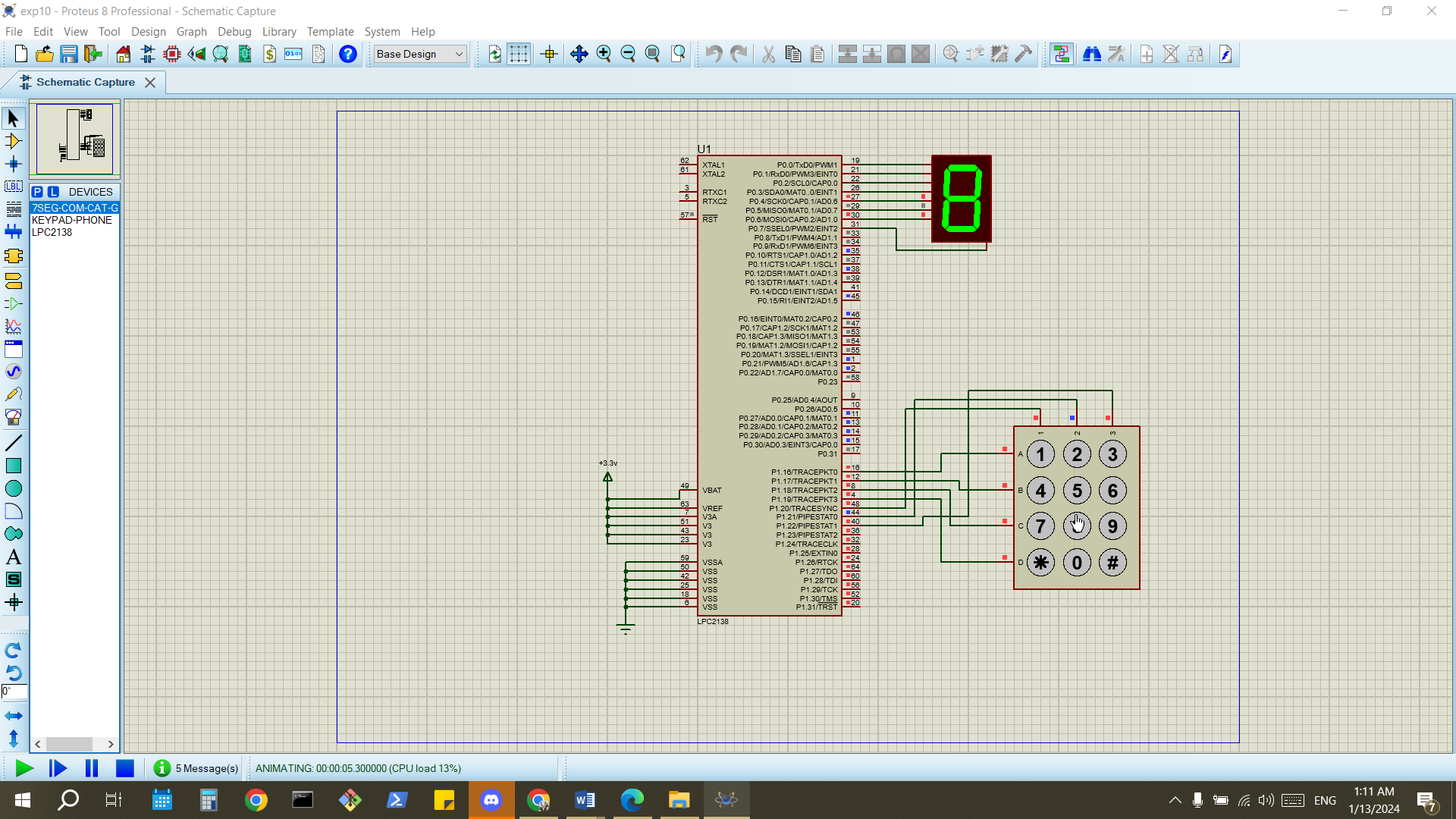
}

### **2.1.2 proteus design**

### **C:\Users\mrabd\Documents\ShareX\Screenshots\2024-01\Chu4ArGDCi.png**

**Fig.2: proteus design for labWork1**

### **2.1.3 results**



**Fig.3: Results for labWork1**

### **2.1.4 result’s Discussion**

This lab work provided a learning experience in microcontroller interfacing, emphasizing the practical aspects of digital signal processing and efficient design in embedded systems.

## **To do task 1**

Edit your program and simulation in order to display the value of the pressed key on a BCD 7-segment. (Binary coded decimal 7-segment display takes the numbers from 0-9 and displays them normally. Its internal work converts the entered decimal to binary by representing the number with four digits then views it as normal 7-seg. At Proteus: BCD display. It has four pins 1-4 from right to left connected to the pins of your port.)

### **code**

#include<lpc213x.h>

#define bit(x) 1<<x

void msDelay(int d) {

int i,j;

long c=0;

d=d\*2;

for(i=0;i<d;i++){

for(j=0;j<1000;j++){

c++;

}

}

}

int dec\_to\_bcd(int number)

{

switch(number){

case 0 : return 0;

case 1 : return 1;

case 2 : return 2;

case 3 : return 3;

case 4 : return 4;

case 5 : return 5;

case 6 : return 6;

case 7 : return 7;

case 8 : return 8;

case 9 : return 9;

default : return 0;

}

}

int main(){

IO0DIR |= 0x0000000F; // P0.0, P0.1, P0.2, and P0.3 as output

IO1DIR &= ~(0x000F0000); //P1.16 ...P1.19 input

IO1DIR |= 0x00700000; // P1.20 ...P1.22 output

while(1){

//test first column

IO1CLR = bit(20);

IO1SET = bit(21)| bit(22);

if(!(IO1PIN & bit(16))){

msDelay(50);

IO0CLR = 0x0000000F;

IO0PIN |= dec\_to\_bcd(1);

}

else if (!(IO1PIN & bit(17))){

msDelay(50);

IO0CLR = 0x0000000F;

IO0PIN |= dec\_to\_bcd(4);

}

else if (!(IO1PIN & bit(18))){

msDelay(50);

IO0CLR = 0x0000000F;

IO0PIN |= dec\_to\_bcd(7);

}

//test second column

IO1CLR = bit(21);

IO1SET = bit(20)| bit(22);

if(!(IO1PIN & bit(16))){

msDelay(50);

IO0CLR = 0x0000000F;

IO0PIN |= dec\_to\_bcd(2);

}

else if (!(IO1PIN & bit(17))){

msDelay(50);

IO0CLR = 0x0000000F;

IO0PIN |= dec\_to\_bcd(5);

}

else if (!(IO1PIN & bit(18))){

msDelay(50);

IO0CLR = 0x0000000F;

IO0PIN |= dec\_to\_bcd(8);

}

else if (!(IO1PIN & bit(19))){

msDelay(50);

IO0CLR = 0x0000000F;

IO0PIN |= dec\_to\_bcd(0);

}

// test third column

IO1CLR = bit(22);

IO1SET = bit(20)| bit(21);

if(!(IO1PIN & bit(16))){

msDelay(50);

IO0CLR = 0x0000000F;

IO0PIN |= dec\_to\_bcd(3);

}

else if (!(IO1PIN & bit(17))){

msDelay(50);

IO0CLR = 0x0000000F;

IO0PIN |= dec\_to\_bcd(6);

}

else if (!(IO1PIN & bit(18))){

msDelay(50);

IO0CLR = 0x0000000F;

IO0PIN |= dec\_to\_bcd(9);

}

}

}

### **proteus design**

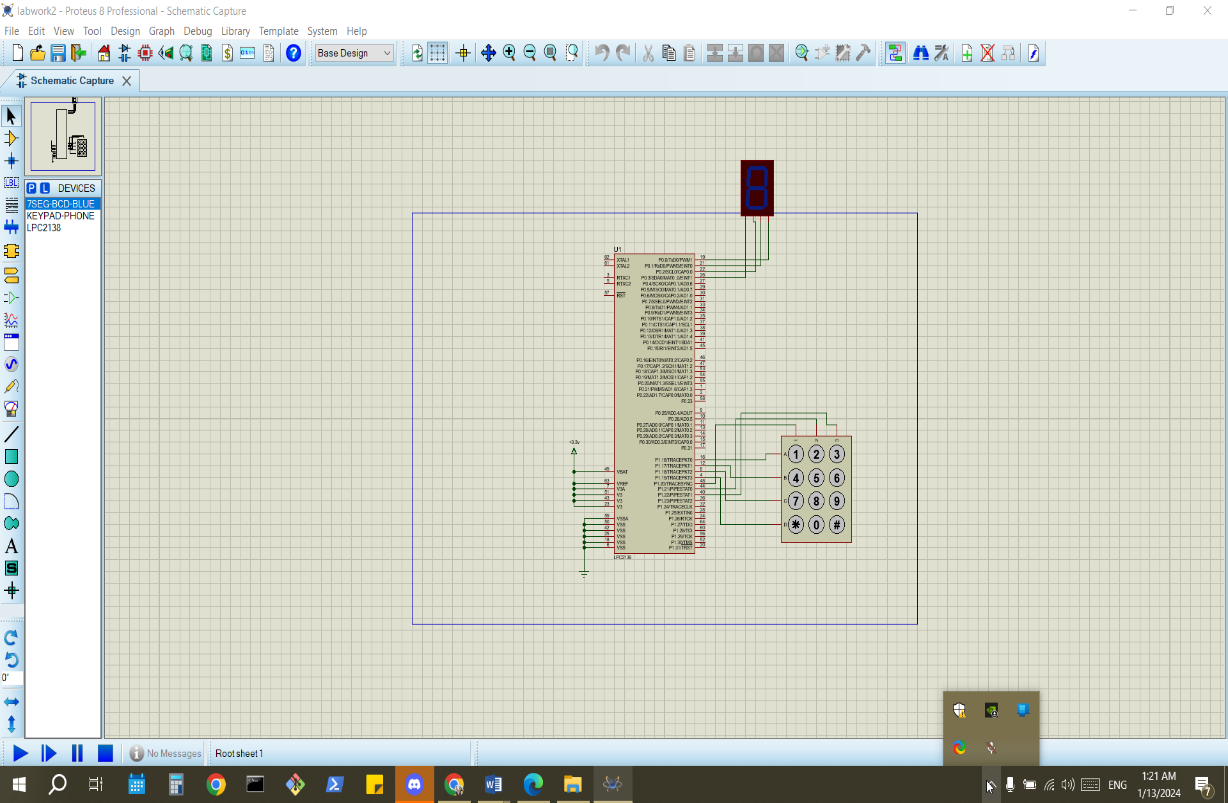


Fig.4: proteus design for to-do 2

### **results**

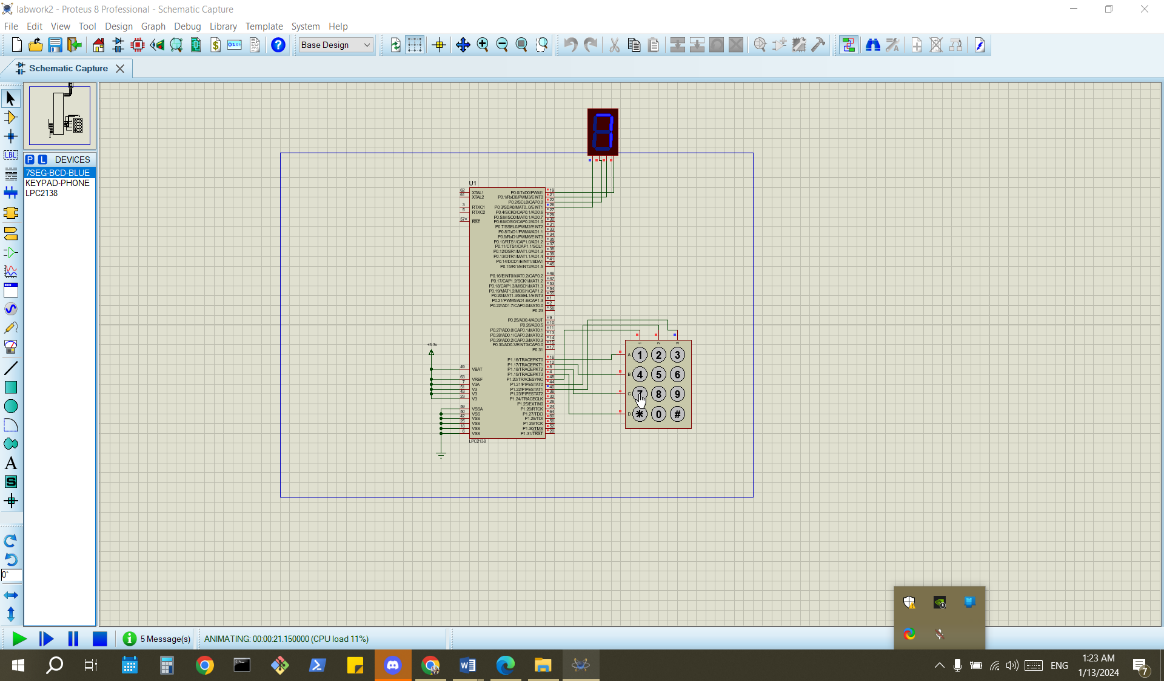


Fig.5: results for to-do 2

### **result’s discussion**

This exercise demonstrated the practical application of matrix configuration for efficient key detection, emphasizing resource optimization in embedded system design but using BCD-7segment display.

## **to-do 2**

### **2.3.1 code**

#include<lpc213x.h>

#define bit(x) 1<<x

const int password[] = {1, 2, 3, 4};

int currentPassword[4];

int passwordIndex = 0;

void msDelay(int d) {

int i,j;

long c=0;

d=d\*2;

for(i=0;i<d;i++){

for(j=0;j<1000;j++){

c++;

}

}

}

int dec\_to\_bcd(int number)

{

return number;

}

int main(){

IO0DIR |= 0x00000001;

IO1DIR &= ~(0x000F0000); //P1.16 ...P1.19 input

IO1DIR |= 0x00700000; // P1.20 ...P1.22 output

while(1){

//test first column

IO1CLR = bit(20);

IO1SET = bit(21)| bit(22);

if(!(IO1PIN & bit(16))){

msDelay(50);

IO0CLR = 0x00000001;

currentPassword[passwordIndex++] = dec\_to\_bcd(1);

}

else if (!(IO1PIN & bit(17))){

msDelay(50);

IO0CLR = 0x00000001;

currentPassword[passwordIndex++] = dec\_to\_bcd(4);

}

else if (!(IO1PIN & bit(19))){

int correct = 1;

int i = 0;

passwordIndex = 0;

for (i = 0 ;i < 4;i= i+1){

if (currentPassword[i] != password[i]){

correct = 0;

break;

}

}

if (correct)

IO0SET = 0x00000001 ; // Set P0.0 high to turn on the LED

}

else if (!(IO1PIN & bit(18))){

msDelay(50);

IO0CLR = 0x00000001;

currentPassword[passwordIndex++]= dec\_to\_bcd(7);

}

//test second column

IO1CLR = bit(21);

IO1SET = bit(20)| bit(22);

if(!(IO1PIN & bit(16))){

msDelay(50);

IO0CLR = 0x00000001;

currentPassword[passwordIndex++] = dec\_to\_bcd(2);

}

else if (!(IO1PIN & bit(17))){

msDelay(50);

IO0CLR = 0x00000001;

currentPassword[passwordIndex++] = dec\_to\_bcd(5);

}

else if (!(IO1PIN & bit(18))){

msDelay(50);

IO0CLR = 0x00000001;

currentPassword[passwordIndex++] = dec\_to\_bcd(8);

}

else if (!(IO1PIN & bit(19))){

msDelay(50);

IO0CLR = 0x00000001;

currentPassword[passwordIndex++] = dec\_to\_bcd(0);

}

// test third column

IO1CLR = bit(22);

IO1SET = bit(20)| bit(21);

if(!(IO1PIN & bit(16))){

msDelay(50);

IO0CLR = 0x00000001;

currentPassword[passwordIndex++] = dec\_to\_bcd(3);

}

else if (!(IO1PIN & bit(17))){

msDelay(50);

IO0CLR = 0x00000001;

currentPassword[passwordIndex++] = dec\_to\_bcd(6);

}

else if (!(IO1PIN & bit(18))){

msDelay(50);

IO0CLR = 0x00000001;

currentPassword[passwordIndex++] = dec\_to\_bcd(9);

}

}

}

### **2.3.2 proteus design**

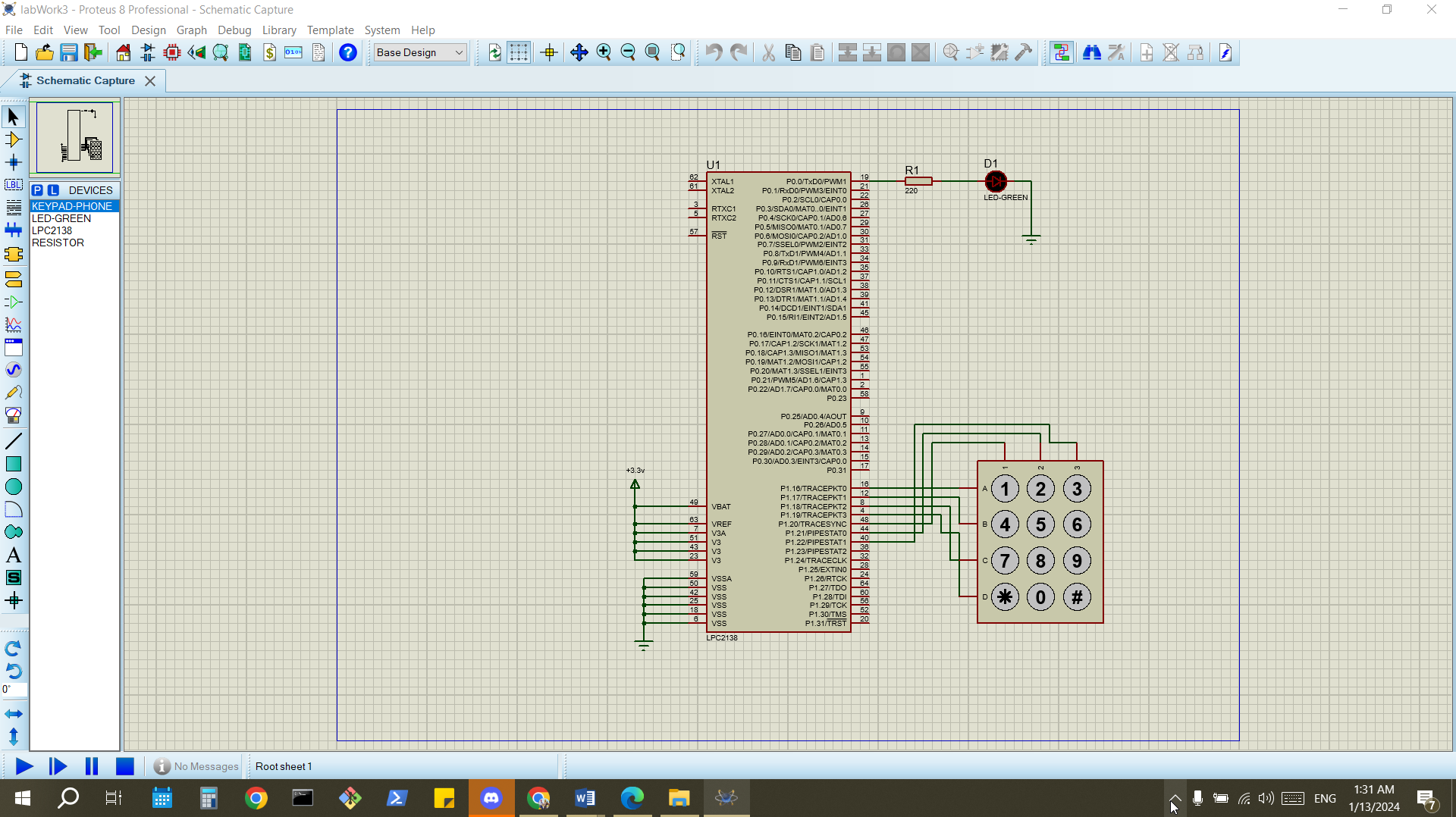


Fig.6: proteus design for to-do 2

### **2.3.3 result**

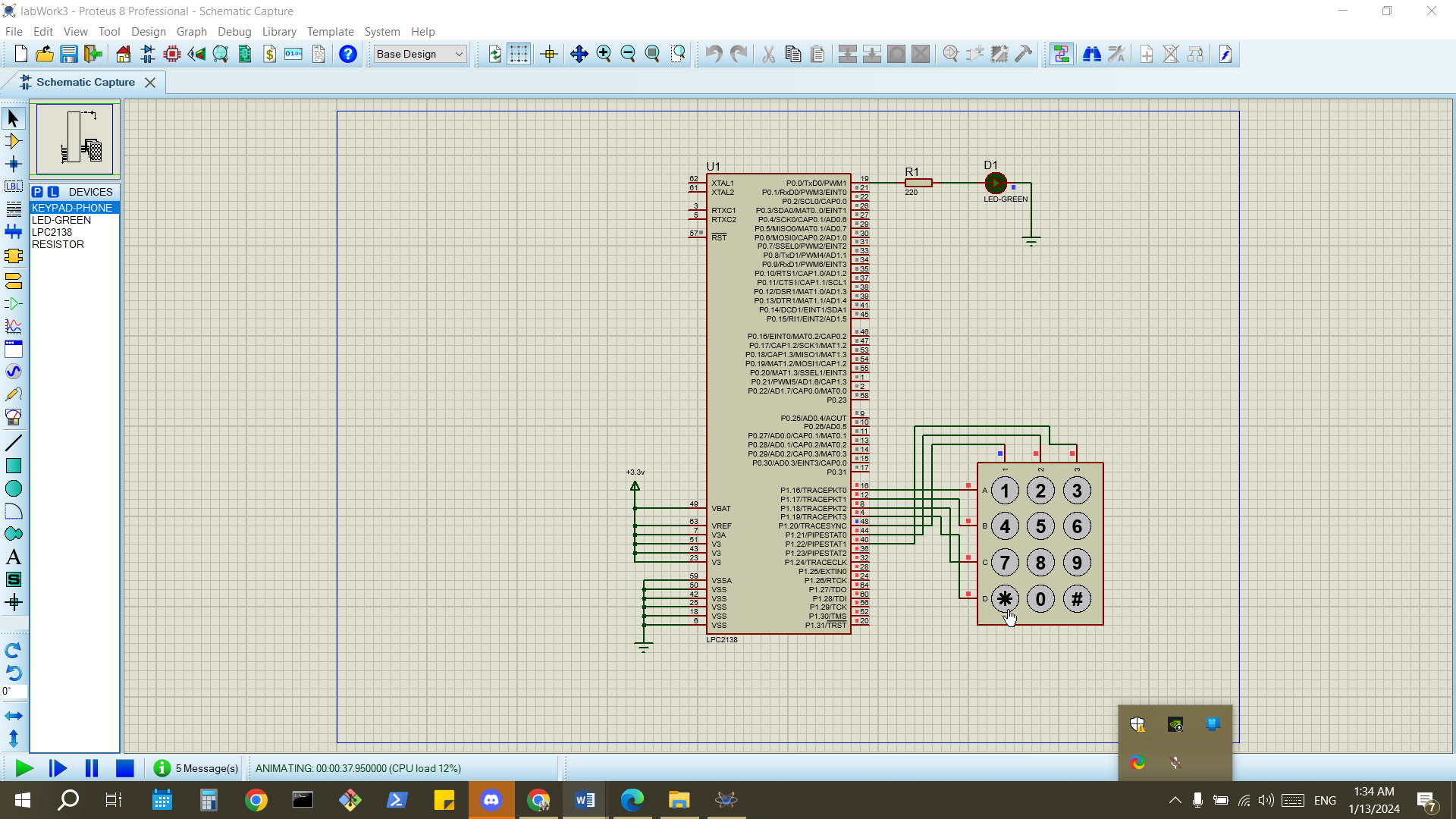


Fig.7: Results for to-do 2

### **2.3.4 result’s discussion**

In To-do 2, the task of creating a password was straightforward, with the '\*' button on the keypad effectively utilized as the confirmation key.

# Conclusion:

The Exp successfully illustrated the principles of keypad interfacing and input handling with a microcontroller. The exercises reinforced key concepts in embedded system design, demonstrating efficient use of hardware resources and practical implementation of user interaction through a matrix keypad.

**References:**

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