B. Tech – Computer and Communication Engineering (CCE)

Third Semester

Amrita School of Engineering

Coimbatore Campus (India)

December 2021

Team 9

S. No	Name	Roll Number
1	Malavika Menon T	CB.EN.U4CCE20031
2	Manoj Parthiban	CB.EN.U4CCE20032
3	Narendran S	CB.EN.U4CCE20036
4	Santosh	CB.EN.U4CCE20053

19CCE201 Microcontrollers and Interfacing Techniques Project

Password-Based Door Open-Lock System Using LPC2148 ARM7 Microcontroller

Faculty In-charge – Peeyush K P Sir

Abstract:

The need for safety can be achieved by making locks that can be electrical or mechanical with one or a few keys, but for locking a big area many locks are required. As everyone knows old fashioned locks are heavyweight and fragile also depending on the tools therefore electronic locks are given more value than mechanical locks.

Nowadays every device's operation is based on digital technology. For example, technology-based identity devices are used for automatic door unlocking or locking. These locking systems are used to control the movement of the door and are functional without requiring a key to lock or unlock the door.

These locking systems are controlled by a keypad and are installed at the side hedge of the door. The main objective of this project is to give safety at every commonplace like home, public places. In this user would give a known password.

The information will be stored in a database. When the correct passcode will be entered, the microcontroller will give instructions to a DC motor. The DC motor will act on door unlocking. Thus, what we want is digital technology to construct an integrated and well-customized safety system at a reasonable price.

Keywords – LPC2148 Microcontroller, LCD, Keypad, Motor

Introduction:

An automatic lock system consists of an electronic control assembly that controls the output load through a password. This output load can be a motor or a lamp or any other mechanical/electrical load.

Here we develop an electronic code lock system using ARM7(LPC2148), which provides control to actuating the load. It is a simple embedded system with input from the keypad and the output being actuated accordingly.

This system demonstrates a password-based door lock system wherein once the correct code or password is entered, the door is opened, and the concerned person is allowed access to the secured area.

Again, if another person arrives it will ask to enter the password. If the password is wrong then the door would remain closed, denying access to the person.

Algorithm:

1. START.
2. Set delay.
3. Define the functions for the DC motor, LCD, and keypad.
4. Set your default password.
5. Define an unsigned char key with a 3x4 matrix to input the keys of the keypad.

6. In main() function:
    1. Call the lcd_int() in the main function.
    2. Set cursor position on the display (cmd(0x80)).
    3. Load the DDRAM in the second line of the LCD to display the letters.
    4. Pass the output of the keypad to be displayed on the LCD.
    5. Display the appropriate command, and call the DC Motor functions for respective operations.
    6. Clear the contents on LCD and set delay.

7. Define function keypad():
    1. Configure (set the direction) the row pins as an output to (P1.16, P1.17, P1.18, P1.19).
    2. Configure (set the direction) the column pins as an output to (P1.20, P1.21, P1.22).
    3. Now ground each row starting from row 0 to 3.
    4. Check the status of row and column simultaneously, if the key is pressed then the respective column will be grounded.
    5. Then the row grounded and column grounded will be known, which implies key press has been detected.
    6. Call the lcd_int() function to display the key pressed on the LCD.

8. Define function lcd_int():

    1. Configure P0.2 & P0.3 pins as output.
    2. Configure RW pin of LCD to ground.
    3. To Enable the 4-bit mode, configure (P0.4, P0.5, P0.6, P0.7) to the data lines D4, D5, D6 and D7 respectively.
    4. Send command on data lines (D4 to D7)
    5. Set RS is low, for the instruction register and EN is low.
    6. Set RS is high for the data register.
    7. Call the function lcd_int() to display the message on line 1 or line 2 of LCD.

9. Define a function for DC Motor:

    1. forward(), configure P0.16 & P0.18 as LOW
    2. When key pressed Set P0.17 as HIGH to rotate the motor in the forward direction.
    3. reverse(), configure P0.17 & P0.18 as LOW
    4. When key pressed Set P0.16 as HIGH to rotate the motor in the reverse direction.
    5. Clear all pins, to stop Motor rotation.

10. END

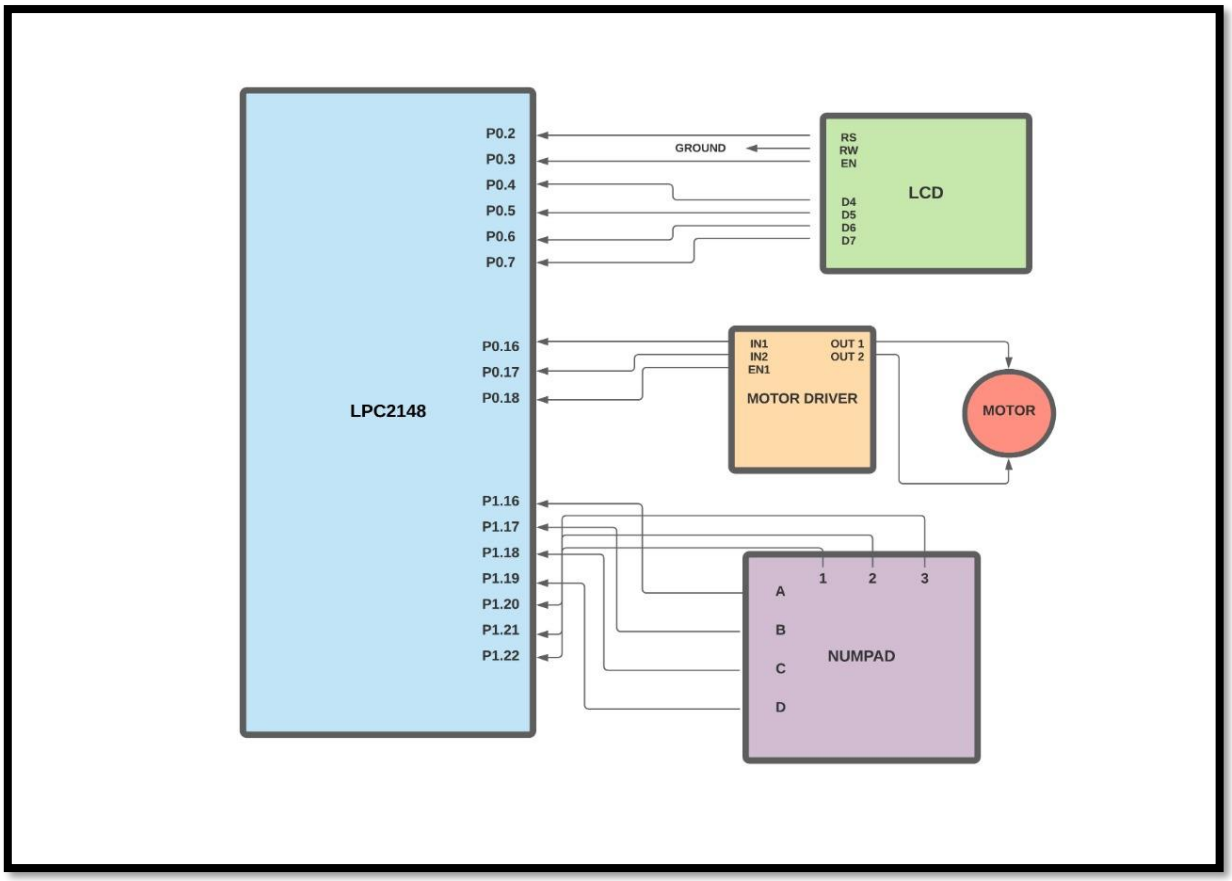


Figure 1 - Block Diagram Representation

Program Code:

```
#include<lpc214x.h>
#define bit(x) (1<<x) // Define a macro (Logic Shift Left)

// Set Default Password:
unsigned char pass[4] = "1234";

// Delay Loop:
void delay ()
{
    unsigned int temp, ct;
    for(ct=0; ct<30; ct++)
    {
        for(temp=0; temp < 65000; temp++);
    }
}

unsigned int range = 0, i; // Variables used for iteration.

// DC Motor Function Definitions:
```

```
void forward(void);
void reverse(void);
void stop(void);

// LCD Function Definitions:
void lcd_init(void);
void cmd(unsigned char a);
void dat(unsigned char b);
void show(unsigned char *s);
void lcd_delay(void);

// Keypad Function Definitions:
#define c1 (IOPIN1&1<<20)
#define c2 (IOPIN1&1<<21)
#define c3 (IOPIN1&1<<22)

// Allocating Row and Column of 4x3 Numeric Keypad:
unsigned char r_loc,c_loc;
unsigned char key[4][3]={"123","456","789","*0#"};
unsigned char keypad(void);

// Main Function:
int main()
{
    unsigned char rx_arr[4];
    int count;

    VPBDIV=0x01; // PCLK = 60 MHz
    IO1DIR |= 0x0f<<16; // P1.16 to P1.19 is assigned for the
numeric 4*3 phone keypad.
    IO0DIR |= 0xf00fc;  // P0.2 to P0.7 is assigned for the
16*2 alphanumeric LCD.

    lcd_init(); // Call the LCD Function

    while(1)
    {
        cmd(0x80); // Set Cursor Position on Display
        show("#Enter Password#");

        cmd(0xc5); // Loading DDRAM to the Second Line of LCD
        for(count=0; count <4; count++)
        {
            rx_arr[count] = keypad(); // Call the Keypad
Function
```

```
            dat('*'); // Password Masking
        }

        if ( (pass[0] == rx_arr[0]) && (pass[1] == rx_arr[1])
&& (pass[2] == rx_arr[2]) && (pass[3] == rx_arr[3]) )
        {
            cmd(0xc0); // Set Cursor Position to the Second
Line
            show("   Thank You!   ");
            forward(); // Motor rotates in forward direction.
            delay(); // Call Delay Function
            stop(); // Motor stops to rotate

            cmd(0xc0); // Set Cursor Position to the Second
Line
            show("  Come Again!!  ");
            delay(); // Call Delay Function
            reverse(); // Motor rotates in reverse direction.
            delay(); // Call Delay Function
            stop(); // Motor stops to rotate
        }

        else
        {
            cmd(0xc0); // Set Cursor Position to the Second
Line
            show("~Wrong Password~");
            delay(); // Call Delay Function
        }

        cmd(0x01); // Contents on the LCD Display are
Cleared
    }
}

// Keypad Functions:
unsigned char keypad()
{
    IO1PIN &= ~(0xff<<16); // P1.16 to P1.23 is initialized to
0.
    IO1PIN |= 0xf0<<16; // P1.16 to P1.19 is configured as
output.

    while(c1 && c2 && c3);
    while(!c1 || !c2 || !c3)
```

```
{
    if(!c1 && c2 && c3)     c_loc=0;
    else if(c1 && !c2 && c3)    c_loc=1;
    else if(c1 && c2 && !c3)    c_loc=2;

    // ROW 1
    IO1CLR = 1<<16; // Make row1 LOW
    IO1SET = 0x0e<<16; // Rest of the rows as '1'
    if(!c1 || !c2 || !c3)
    {
        // Scan for Keypress:
        r_loc=0;
        break;
    }

    // ROW 2
    IO1CLR = 1<<17; // Make row2 LOW
    IO1SET = 0x0d<<16; // Rest of the rows as '1'
    if(!c1 || !c2 || !c3)
    {
        // Scan for Keypress:
        r_loc=1;
        break;
    }

    // ROW 3
    IO1CLR = 1<<18; // Make row3 LOW
    IO1SET = 0x0b<<16; // Rest of the rows as '1'
    if(!c1 || !c2 || !c3)
    {
        // Scan for Keypress:
        r_loc=2;
        break;
    }

    // ROW 4
    IO1CLR = 1<<19; // Make row4 LOW
    IO1SET = 0x07<<16; // Rest of the rows as '1'
    if(!c1 || !c2 || !c3)
    {
        // Scan for Keypress:
        r_loc=3;
        break;
    }
}
```

```
    while(!c1 || !c2 || !c3);
    return (key[r_loc][c_loc]); // Return Value to Display
}

// LCD Functions:

// Function to Configure LCD:
void lcd_init()
{
    cmd(0x02); // Initialize cursor to home position
    cmd(0x28); // Set the Function Set (4-bit interface, two
line, 5X8 dots)
    cmd(0x0c); // Display ON, Cursor OFF, & Blink OFF
    cmd(0x06); // Set the Entry Mode (Cursor Increment, Display
Shift OFF)
    cmd(0x80); // Set DDRAM or cursor position on display
}

// Display Character Using LCD Command Function:
void cmd(unsigned char a)
{
    IO0PIN &= 0xffffff03; // Set P0.2 - P0.7 as input
    IO0PIN |= (a & 0xf0) << 0; // Pass the MSB, Most
Significant Bit (last 4 bits of our input 8-bit data into input
pins)
    IO0CLR |= bit(2); // Command Input Method Instruction, Set
Register Select to 0 (RS = 0)
    IO0CLR |= bit(1); // Write to LCD, Set Read/Write to 0 (RW
= 0)
    IO0SET |= bit(3); // LCD will be enabled ON until delay is
performed (EN = 1)
    lcd_delay(); // Call LCD Delay Function
    IO0CLR |= bit(3); // LCD will be turned OFF, EN = 0
(Ground)

    // Same procedure followed again, for LSB Bit:
    IO0PIN &= 0xffffff03; // Set P0.2 - P0.7 as input
    IO0PIN |= ((a << 4)  & 0xf0) << 0; // Pass the LSB, Least
Significant Bit (last 4 bits of our input 8-bit data into input
pins)
    IO0CLR |= bit(2); // Command Input Method Instruction, Set
Register Select to 0 (RS = 0)
    IO0CLR |= bit(1); // Write to LCD, Set Read/Write to 0 (RW
= 0)
```

```
    IO0SET |= bit(3); // LCD will be enabled ON until delay is
performed (EN = 1)
    lcd_delay(); // Call LCD Delay Function
    IO0CLR |= bit(3); // LCD will be turned OFF, EN = 0
(Ground)
}

// Display Character Using Data Command Function:
void dat(unsigned char b)
{
    IO0PIN &= 0xffffff03; // Set P0.2 - P0.7 as input
    IO0PIN |= (b & 0xf0) << 0; // Pass the MSB, Most
Significant Bit (last 4 bits of our input 8-bit data into input
pins)
    IO0SET |= bit(2); // Data Input Method Instruction, Set
Register Select to 1 (RS = 1)
    IO0CLR |= bit(1); // Write to LCD, Set Read/Write to 0 (RW
= 0)
    IO0SET |= bit(3); // LCD will be enabled ON until delay is
performed (EN = 1)
    lcd_delay(); // Call LCD Delay Function
    IO0CLR |= bit(3); // LCD will be turned OFF, EN = 0
(Ground)

    // Same procedure followed again, for LSB Bit:
    IO0PIN &= 0xffffff03; // Set P0.2 - P0.7 as input
    IO0PIN |= ((b << 4)  & 0xf0) << 0; // Pass the LSB, Least
Significant Bit (last 4 bits of our input 8-bit data into input
pins)
    IO0SET |= bit(2); // Data Input Method Instruction, Set
Register Select to 1 (RS = 1)
    IO0CLR |= bit(1); // Write to LCD, Set Read/Write to 0 (RW
= 0)
    IO0SET |= bit(3); // LCD will be enabled ON until delay is
performed (EN = 1)
    lcd_delay(); // Call LCD Delay Function
    IO0CLR |= bit(3); // LCD will be turned OFF, EN = 0
(Ground)
}

// We have involved the concept of password masking here:
void show(unsigned char *s)
{
```

```
    while(*s) // It involves a text field that accepts any
character, however, doesn't show the inputted character to the
user.
    {
        dat(*s++); // Instead, it shows an asterisk.
    }
}

// LCD Delay Function Loop:
void lcd_delay()
{
    unsigned int i;
    for(i=0;i<=1000;i++);
}

// DC Motor Functions:
// Here, we have the following three pins - Input 1 Pin
configuring P0.16, Input 2 Pin configuring P0.17 and Enable Pin
configuring P0.18.
void forward()
{
    IO0SET = bit(16) | bit(18); // Configure P0.16 and P0.18 as
LOW
    IO0CLR = bit(17); // Motor rotates in forward direction.
}

void reverse()
{
    IO0SET = bit(17) | bit(18); // Configure P0.17 & P0.18 as
LOW
    IO0CLR = bit(16); // Motor rotates in reverse direction.
}

void stop()
{
    IO0CLR = bit(18); // When P0.18 is cleared, motor rotation
stops.
}
```

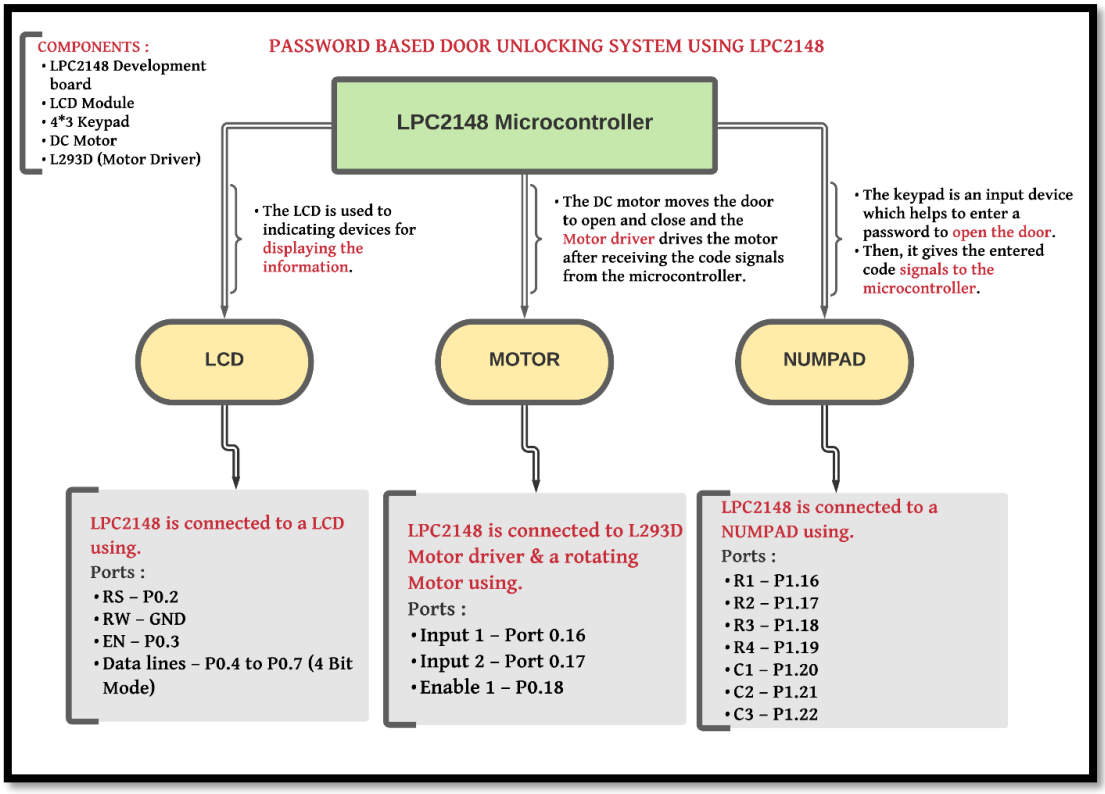


Figure 2 - Flowchart Representation of Project

Simulation Results:

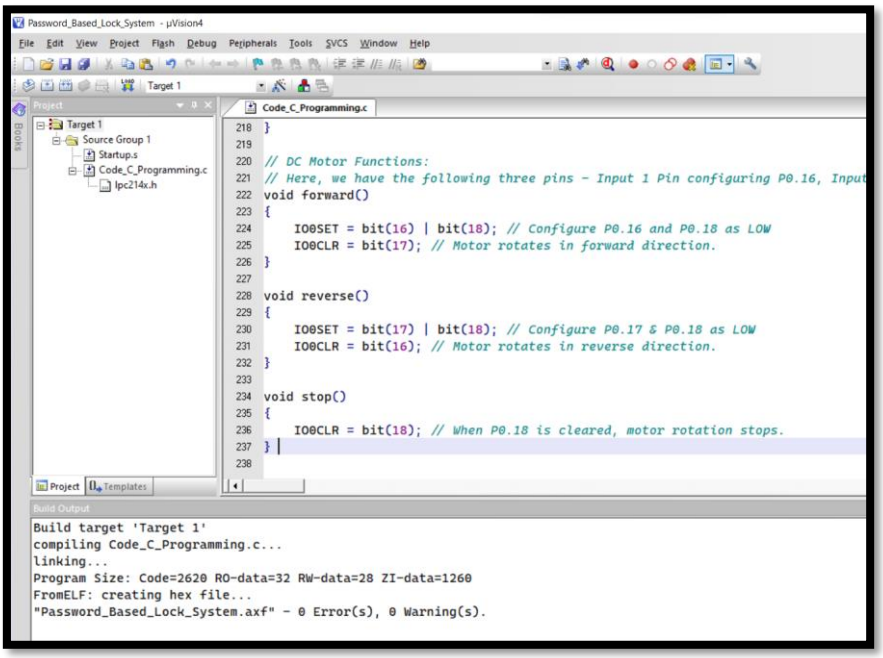
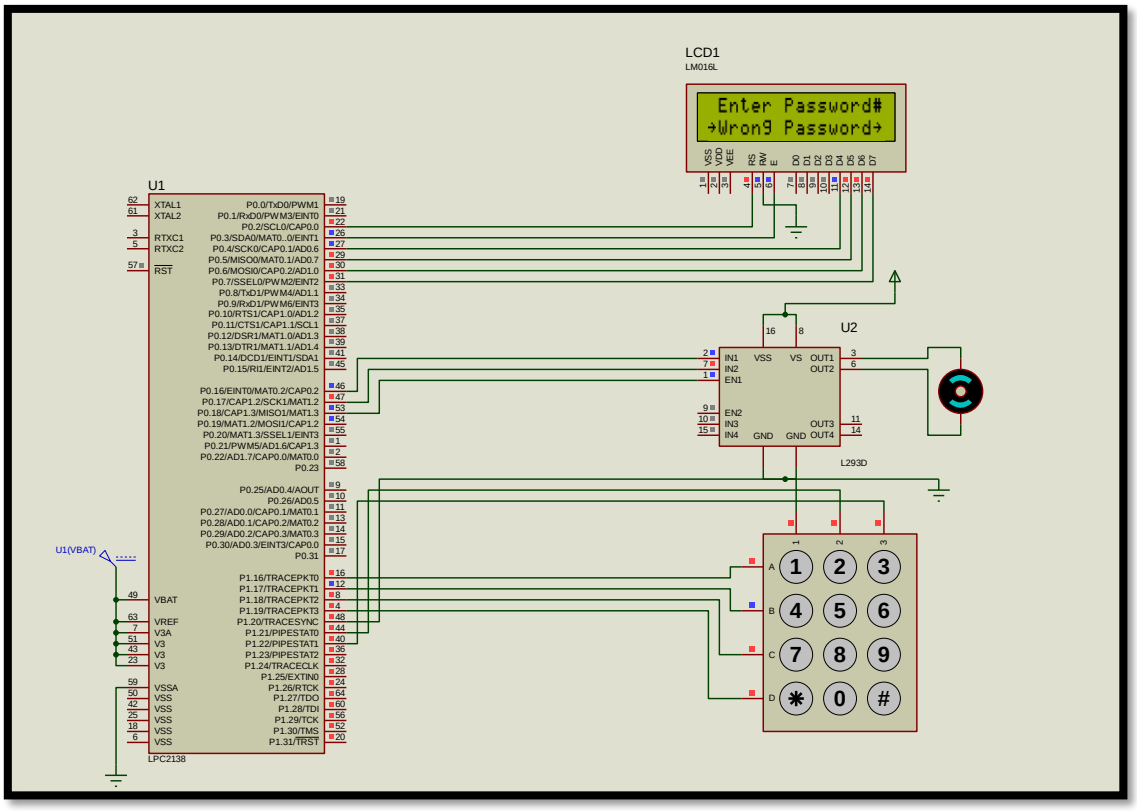


Figure 3 - Built Keil µVision4 Code



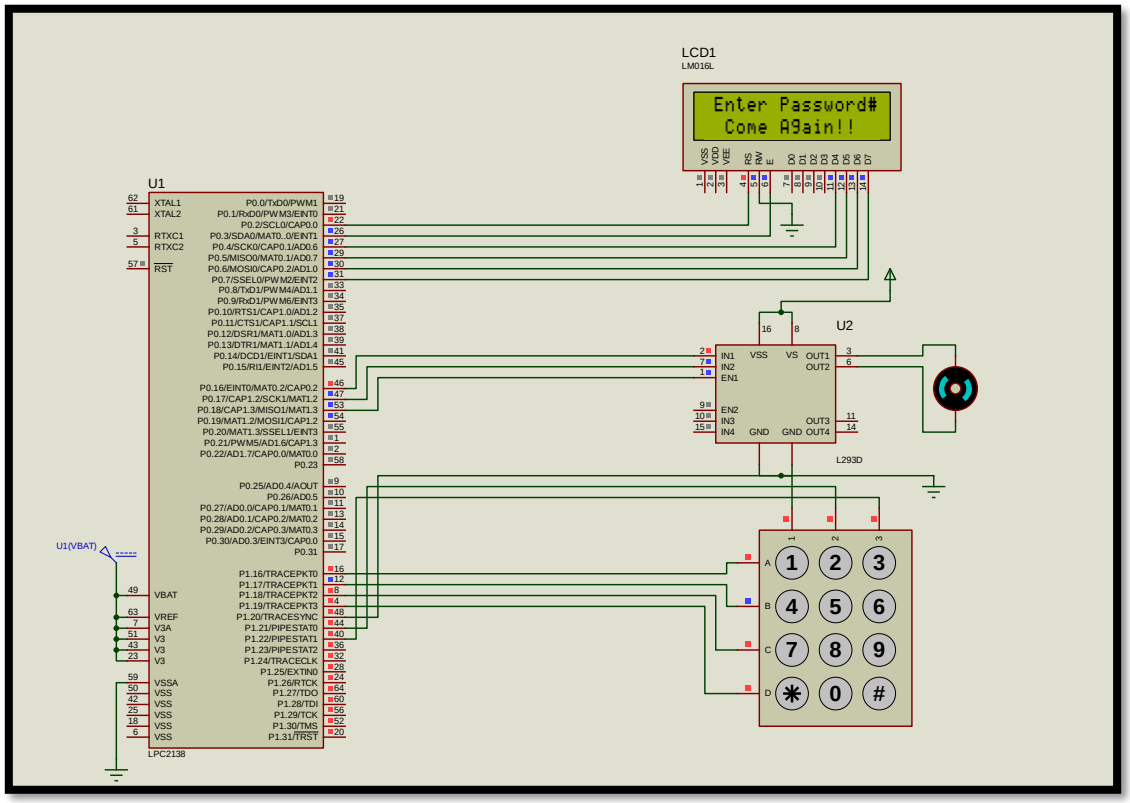


Figure 4 – Proteus 7 Professional Simulation

Conclusion and Future Prospects:

The security level can be increased by adding a biometric fingerprint scanner. We can interface sensors like Fire, LPG, PIR motion detectors to the microcontroller in case of an accident so that door will open automatically.

We can interface a camera to the microcontroller so that it could capture the picture of the thief who is trying to breach security. This simple circuit can be used at places like home to ensure better safety. With a slight modification, this project can also be used to control the switching of loads through a password.

It can also be used by organizations to ensure authorized access to highly secured places. This project is productive in providing enough security if the password is not shared.

In future, this "Password-Based Door Open-Lock System" can be provided maximum security by the above enhancements to completely satisfy the user's needs. Hence, a common man can afford to buy such a locking system at minimal cost to keep his valuables safely without any worries.

References:

1. Password-Based Door Lock System, International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, Feb 2019 (www.irjet.net)
2. Embedded High Alert Door Locking System Using IoT Technology, International Journal of Computer Science and Engineering (IJCSE) Special Issue, May 2017 (https://www.internationaljournalssrg.org/uploads/specialissuepdf/ICACCE/2017/CSE/IJCSE-NCACCE-P104.pdf)
3. Onetime Password Based Door Lock System: https://hackprojects.wordpress.com/projects/arm-projects/password-and-onetime-password-based-door-lock-system/
4. Password-Based Door Open System Using LPC2148: https://embetronicx.com/projects/lpc2148-projects/password-based-door-open-system-using-lpc2148/
5. YouTube LPC2148 Microcontroller Tutorials.