# Application Development using ARM Cortex M micro-controller

# Password Based Smart Door Lock

Team member 1: Tanay Aggarwal
Reg.No: 230911278
Department of I&CT
Manipal Institute of Technology
Manipal Academy of Higher Education
Manipal- 576104, Karnataka, India
Email: tanay.mitmpl2023@learner.manipal.edu

Team member 2: Raghav Khetarpal
Reg.No: 230911266
Department of I&CT
Manipal Institute of Technology
Manipal Academy of Higher Education
Manipal- 576104, Karnataka, India
Email: raghav3.mitmpl2023@learner.manipal.edu

Team member 3: Srivastava Prakhar
Reg.No: 230911246
Department of I&CT
Manipal Institute of Technology
Manipal Academy of Higher Education
Manipal- 576104, Karnataka, India
Email:prakhar2.mitmpl2023@learner.manipal.edu

**Abstract:**
**This study presents the development of a password-based smart door lock system, leveraging modern technology to enhance access control and security measures. The system utilizes a microcontroller as its core component, interfacing with a numeric keypad, a stepper motor, and an LCD display. Users input a password via the keypad, which is then verified against a predefined one stored within the microcontroller's memory. Upon successful authentication, the microcontroller commands a stepper motor to unlock the door, while providing real-time feedback on the lock status through the LCD display. This project offers an efficient and secure solution for implementing a door lock system with passwordbased authentication, demonstrating the potential of smart technologies in enhancing access control in various environments.**

## I. INTRODUCTION

This project focuses on developing a password-based door lock system using a microcontroller to interface with a numeric keypad, a stepper motor, and an LCD display. The microcontroller serves as the brain of the system, receiving user input for a password via the numeric keypad, validating it against a predefined password, and controlling a lock mechanism using a stepper motor to unlock or lock a door based on the authentication result. The status of the lock, whether it is unlocked or locked, is displayed on the LCD display, providing visual feedback to the user. The project offers an efficient and secure way to implement a door lock system with password-based authentication, leveraging the capabilities of a microcontroller for interfacing with peripherals and ensuring the security of access to a restricted area.

## II. PROBLEM STATEMENT & OBJECTIVE

Problem Statement: The need for secure access control to restricted areas has become increasingly important in various applications such as homes, offices, and facilities. Traditional mechanical lock and key systems may not provide adequate security, and the management of physical keys can be cumbersome. Therefore, there is a demand for a password-based door lock system that offers efficient and secure access control. Existing password-based door lock systems may be expensive or complex, requiring external controllers or software. There is a need for a simplified and cost-effective solution that utilizes a microcontroller to interface with a numeric keypad and an LCD display for password entry and status indication, respectively.

Objective: The objective of this project is to develop a reliable and secure password-based door lock system using a microcontroller, numeric keypad, stepper motor, and LCD display. The system will receive user input for a password, validate it against a predefined password, and control a lock mechanism to unlock or lock the door. Additionally, the system will provide visual feedback on the lock status through an LCD display. The design will be user-friendly, cost-effective, and suitable for various applications, while ensuring robust security and reliable operation. The goal is to provide an efficient and secure access control solution for restricted areas.

## III. CONFIGURATIONS & ASSUMPTIONS

Configurations:

•LCD(CND): data- P0.23 to P0.26 RS -P0.27 En -P0.28

•Motor (PIN1): P0.8 to P0.11

•Buzzer(CNC):P0.15

•Keypad (CNB): Input (Column)-P1.23 to P1.26 Output (Row)-P2.10 to P2.13

Assumptions:

•The password is predefined.

•The password input can be any number between 1 to 9 with some special characters and is of 4 digits.

•The door can only be opened from inside without the lock and needs the pin when you enter from outside.

## IV. WORKING WITH BLOCK DIAGRAM

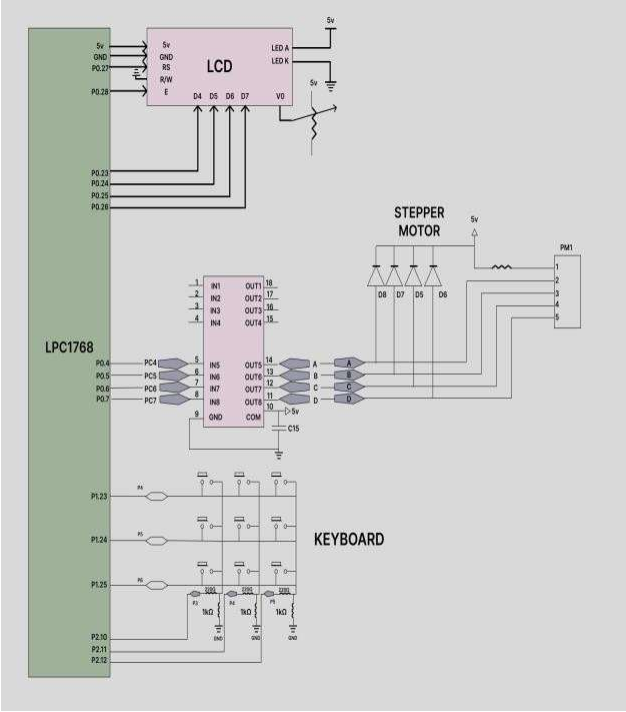


Figure 1: Circuit Diagram

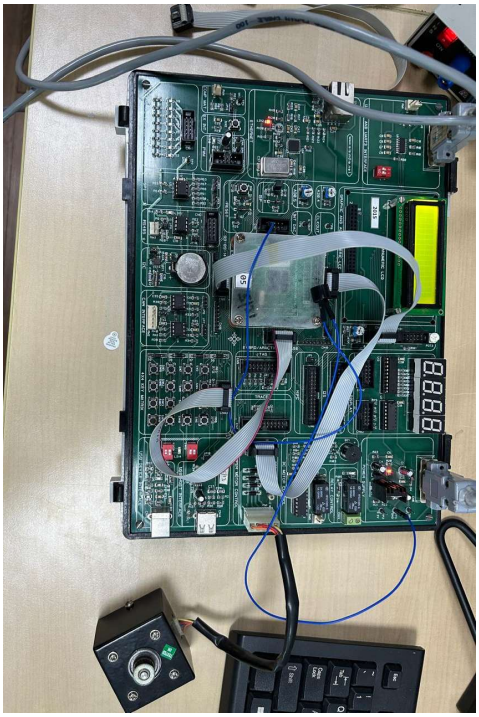

Figure 2: Implementation with needed components

Working Explanation –

1. LCD Initialization:
   - o The program initializes a 16x2 LCD display connected to the LPC1768 microcontroller.
   - o The lcd_write, port_write, and related functions handle the communication with the LCD.
2. Buzzer Initialization:

o The initBuzzer, turnOnBuzzer, and turnOffBuzzer functions are responsible for setting up and controlling a buzzer connected to pin P0.15.

3. Keypad Scanning:

o The code scans a 4x4 matrix keypad to detect key presses.

o The outer loop continuously scans for key presses until all four digits of a password are entered.

4. Password Authentication:

o The entered password is compared with a predefined password (pass array).

- If the password is correct, a success message is displayed on the LCD, and the motor is rotated in both clockwise and anticlockwise directions, simulating the opening of a door.

4. LCD Display:

o The LCD is used to display messages, including successful authentication and incorrect password attempts.

6. Motor Control:

o The clockwise and anticlockwise functions control the motor by setting specific patterns on the GPIO pins (P0.8 to 0.11).

7. Buzzer Feedback:

o The buzzer is used to provide feedback on incorrect password attempts. It is turned on and off using the turnOnBuzzer and turnOffBuzzer functions.

8. Timeout Mechanism:

o There is a timeout mechanism (timeout variable) that increases exponentially on each incorrect password attempt.

9. Infinite Loop:

o The program runs in an infinite loop, continually scanning for keypad input and responding accordingly.

V. PROGRAM

```
#include <lpc17xx.h>

#define RS_CTRL 0x08000000 // P0.27, 1<<27
#define EN_CTRL 0x10000000 // P0.28, 1<<28
#define DT_CTRL 0x07800000 // P0.23 to P0.26 data lines, F<<23
#define BUZZER_PIN (1 << 15)

unsigned long int temp1 = 0, temp2 = 0, i, j, r, k, clkw[4] = {0x6, 0xc, 0x9, 0x3}, anticlkw[4] = {0x3, 0x9, 0xc, 0x6}, timeout = 10000000;
unsigned char flag1 = 0, flag2 = 0, key;
int pass[4] = {7, 4, 6, 2};
unsigned char msg[] = {"authenticated successfully"};
unsigned char msg2[] = {"incorrect password"};
unsigned char msg3[] = {"incorrect password"};
unsigned int row, col, flag, x;
int a, b, c, d, count = 0, iflag = 0, ans, idx = 1;
```

```
16. char op;
17. 
18. unsigned long int var1,var2;
19. 
20. void lcd_write(void);
21. void port_write(void);
22. void delay_lcd(unsigned int);
23. void clockwise(void);
24. void delay(unsigned long int);
25. void anticlockwise(void);
26. void initBuzzer(void);
27. void turnOnBuzzer(void);
28. void turnOffBuzzer(void);
29. 
30. unsigned long int init_command[10];
31. unsigned char ascii[4][4] = {{'0', '1', '2', '3'}, {'4', '5', '6', '7'}, {'8', '9', '+', '-'}, {'!', '@', '#',
'$'}};
32. 
33. int main(void)
34. {
35.     SystemInit();
36.     SystemCoreClockUpdate();
37. 
38.     LPC_PINCON->PINSEL0 &= ~(0x3F << 12);
39.     LPC_PINCON->PINSEL0 &= ~(0x3 << 30);//added p0.15 as gpio pin
40.     LPC_PINCON->PINSEL0 &= ~(0xF << 8);//added 0.4 to 0.7 as motor
41.     LPC_PINCON->PINSEL1 &= ~((0x3 << 18) | (0x3 << 22));
42.     LPC_GPIO0->FIODIR |= DT_CTRL | RS_CTRL | EN_CTRL; // 0xf<<23 | 1<<27 | 1<<28;
43.     LPC_GPIO2->FIODIR |= 0XF << 10;
44.     LPC_GPIO1->FIODIR &= ~0xF << 23;
45.     LPC_GPIO0->FIODIR |= 0xF;
46. LPC_GPIO0->FIODIR |= 0xF << 4;
47. 
48.     initBuzzer();
49.     flag1 = 0;
50. 
51.     init_command[0] = 0x30;
52.     init_command[1] = 0x30;
53.     init_command[2] = 0x30;
54.     init_command[3] = 0x20;
55.     init_command[4] = 0x28;
56.     init_command[5] = 0x01;
57.     init_command[6] = 0x06;
58.     init_command[7] = 0x0c;
59.     init_command[8] = 0x80;
60. 
61.     for (i = 0; i < 9; i++)
62.     {
63.         temp1 = init_command[i];
64.         lcd_write();
65.     }
66.     flag1 = 1; // DATA MODE
67.     while (1)
68.     {
69.         if (iflag == 4)
70.         {
71.             temp1 = 0x01;
72.             flag1 = 0;
73.             lcd_write();
74.             temp1 = 0x80;
75.             lcd_write();
76.             flag1 = 1;
77.             i = 0;
78.             while (msg3[i] != '\0')
79.             {
80.                 temp1 = msg3[i];
81.                 lcd_write();
82.                 i += 1;
83.                 if (i == 16) // check for 1 characters in first line
84.                 {
85.                     flag1 = 0;    // if yes
```

```
                    temp1 = 0xc0; // configure second line in command register
                    lcd_write();
                    flag1 = 1;
                }
            }
            turnOnBuzzer();
            delay(timeout);
            turnOffBuzzer();
            timeout = timeout * 2;
            iflag = 0;
            temp1 = 0x01;
            flag1 = 0;
            lcd_write();
            temp1 = 0x80;
            lcd_write();
            flag1 = 1;
        }

        while (count < 4)
        {

            for (row = 0; row < 4; row++)
            {
                flag = 0;
                LPC_GPIO2->FIOPIN = 1 << (10 + row); // sending high on ith row
                for (j = 0; j < 100; j++)
                    ;

                x = ((LPC_GPIO1->FIOPIN >> 23) & 0xf); // CHECK IF ANY COLUMN LINE IS HIGH

                if (x)
                {

                    // some key is pressed
                    if (x == 1) // finding col no.
                        col = 0;
                    else if (x == 2)
                        col = 1;
                    else if (x == 4)
                        col = 2;
                    else if (x == 8)
                        col = 3;
                    count++;
                    key = ascii[row][col];
                    temp1 = key;
                    lcd_write();
                    if (count == 1)
                    {
                        a = key - 0x30;
                    }
                    else if (count == 2)
                    {
                        b = key - 0x30;
                    }
                    else if (count == 3)
                    {
                        c = key - 0x30;
                    }
                    else if (count == 4)
                    {
                        d = key - 0x30;
                    }
                }
                for (i = 0; i < 500000; i++)
                    ;
            }
        }
        temp1 = 0x01;
        flag1 = 0;
        lcd_write();
        temp1 = 0x80;
```

```
        lcd_write();
        flag1 = 1;
        if (a == pass[0] && b == pass[1] && c == pass[2] && d == pass[3])
        {
                                                    iflag=0;
            i = 0;
            while (msg[i] != '\0')
            {
                temp1 = msg[i];
                lcd_write();
                i += 1;
                if (i == 16) // check for 1 characters in first line
                {
                    flag1 = 0;    // if yes
                    temp1 = 0xc0; // configure second line in command register
                    lcd_write();
                    flag1 = 1;
                }
            }
            for (j = 0; j < 70    ; j++)
            {
                clockwise(); // Rotate the motor clockwise to open the door
            }
            delay(650000);
            for (j = 0; j < 70; j++)
            {
                anticlockwise();
            }
        LPC_GPIO0 -> FIOCLR = 0xF << 4;
        }
        else
        {
               a=0;
               b=0;
               c=0;
               d=0;
               count=0;
            iflag++;

        }
    }
}

void initBuzzer(void)
{
    // Set P0.15 as an output for the buzzer
    LPC_GPIO0->FIODIR |= BUZZER_PIN;
}

void turnOnBuzzer(void)
{
    // Turn on the buzzer by setting P0.15 high
    LPC_GPIO0->FIOSET |= BUZZER_PIN;
}

void turnOffBuzzer(void)
{
    // Turn off the buzzer by setting P0.15 low
    LPC_GPIO0->FIOCLR |= BUZZER_PIN;
}

void clockwise(void)
{
    for (i = 0; i < 4; i++)
    {
        LPC_GPIO0->FIOCLR = 0xF << 4;               // Clear previous output on P0.4-P0.7
        LPC_GPIO0->FIOSET = clkw[i] << 4;           // Output step pattern
        for (k = 0; k < 5000; k++);                 // Delay
    }
}

```

```
228.
229. void delay(unsigned long int n)
230. {
231.     for (k = 0; k < n; k++)
232.         ;
233. }
234.
235. void anticlockwise(void)
236. {
237.     for (i = 0; i < 4; i++)
238.     {
239.         LPC_GPIO0->FIOCLR = 0xF << 4;
240.         LPC_GPIO0->FIOSET = anticlkw[i] << 4;
241.         for (k = 0; k < 5000; k++);
242.     }
243. }
244.
245.
246. void lcd_write(void)
247. {
248.     temp2 = temp1 & 0xf0; // 4 - Bits to get it to least significant digit place
249.     temp2 = temp2 >> 4;
250.     port_write();
251.     if (!((flag1 == 0) && ((temp1 == 0x20) || (temp1 == 0x30)))) // send least significant 4 bits only when it is
data/command other than 0x30/0x20
252.     {
253.         temp2 = temp1 & 0x0f;
254.         temp2 = temp2;
255.         port_write();
256.     }
257. }
258.
259. void port_write(void)
260. {
261.     LPC_GPIO0->FIOPIN = temp2 << 23; // sending the ascii code
262.     if (flag1 == 0)
263.         LPC_GPIO0->FIOCLR = RS_CTRL; // if command
264.     else
265.         LPC_GPIO0->FIOSET = RS_CTRL; // if data
266.
267.     LPC_GPIO0->FIOSET = EN_CTRL; // sending a low high edge on enable input
268.     for (r = 0; r < 25; r++)
269.         ;
270.     LPC_GPIO0->FIOCLR = EN_CTRL;
271.     for (r = 0; r < 30000; r++)
272.         ;
273. }
```

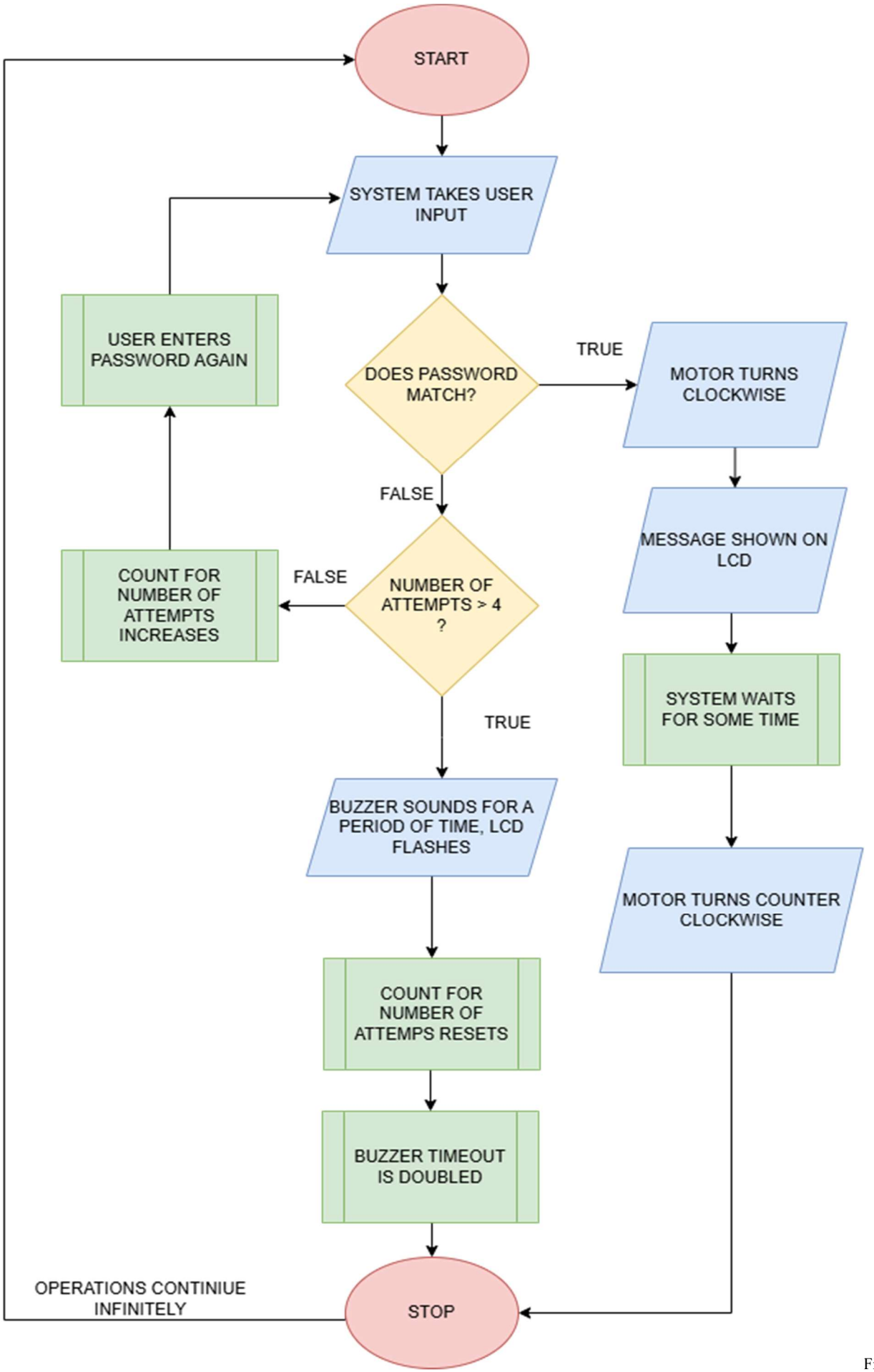


Figure 3: Workflow

REFERENCES:

[1] Aruna Kommu, Naveen Kumar Uttakar, Raghavendra Rao Kanchi. “Design and development of sensor-based mini projects for embedded system laboratory using ARM Cortex-M3 (LPC1768)”