MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Mini Project Report
of
Embedded Systems (CSE)

Nokia-Phone Style Keypad

SUBMITTED
BY

Jithin Joseph , 210905372 , 58
Abel Koshy , 210905029 , 6
Meghana Ganesh , 210905160 , 26
Sujeet Amberkar , 200905092
Ronit Saini , 210905322 , 51

Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal.

April 2023

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal
22/04/2023

CERTIFICATE

This is to certify that the project titled **Nokia-Phone Style Keypad** is a record of the bonafide work done by **Jithin Joseph (210905372), Abel Koshy (210905029), Meghana Ganesh (210905160), Sujeet Amberkar (200905092) and Ronit Saini(210905322)** submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in COMPUTER SCIENCE & ENGINEERING of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2022-2023.

Name and Signature of Examiners:

TABLE OF CONTENTS

ABSTRACT

This project aims to design and implement a keyboard interface for microcontrollers that emulates the functionality of phone keyboards. The interface should enable users to input all letters from A to Z and all numbers from 0 to 9. The project involves designing and developing the hardware and software components required to enable this functionality. The end result will be a user-friendly keyboard interface that provides an efficient and reliable means of inputting characters into microcontroller-based systems.

CHAPTER 1: OBJECTIVE

1. To design and develop a software driver that can be used to read input from the keyboard interface.
2. To enable the keyboard interface to recognize and read all letters from A to Z and all numbers from 0 to 9.
3. To integrate the keyboard interface with microcontroller-based systems and demonstrate its functionality.
4. To document the design, development, and testing process.

CHAPTER 2: ASSUMPTIONS

1. The only available characters are A-Z and 0-9
2. The equipment functions properly as intended.
3. The project aims to display the intended outputs when keys are pressed on the keypad.
4. The keyboard matrix behaves similar to that of a phone keypad where a single key on a matrix can output more than two values.
   Eg: If key 1 can output 4 values that are A,B,C,1 then a single press of key 1 would give A as output, 2 presses gives B and so on.

5. Special character (#) is utilized for indicating the end of a word or a string.
6. Special character (_) is utilized for indicating the end of key presses.

CHAPTER 3: CONFIGURATIONS

- LCD

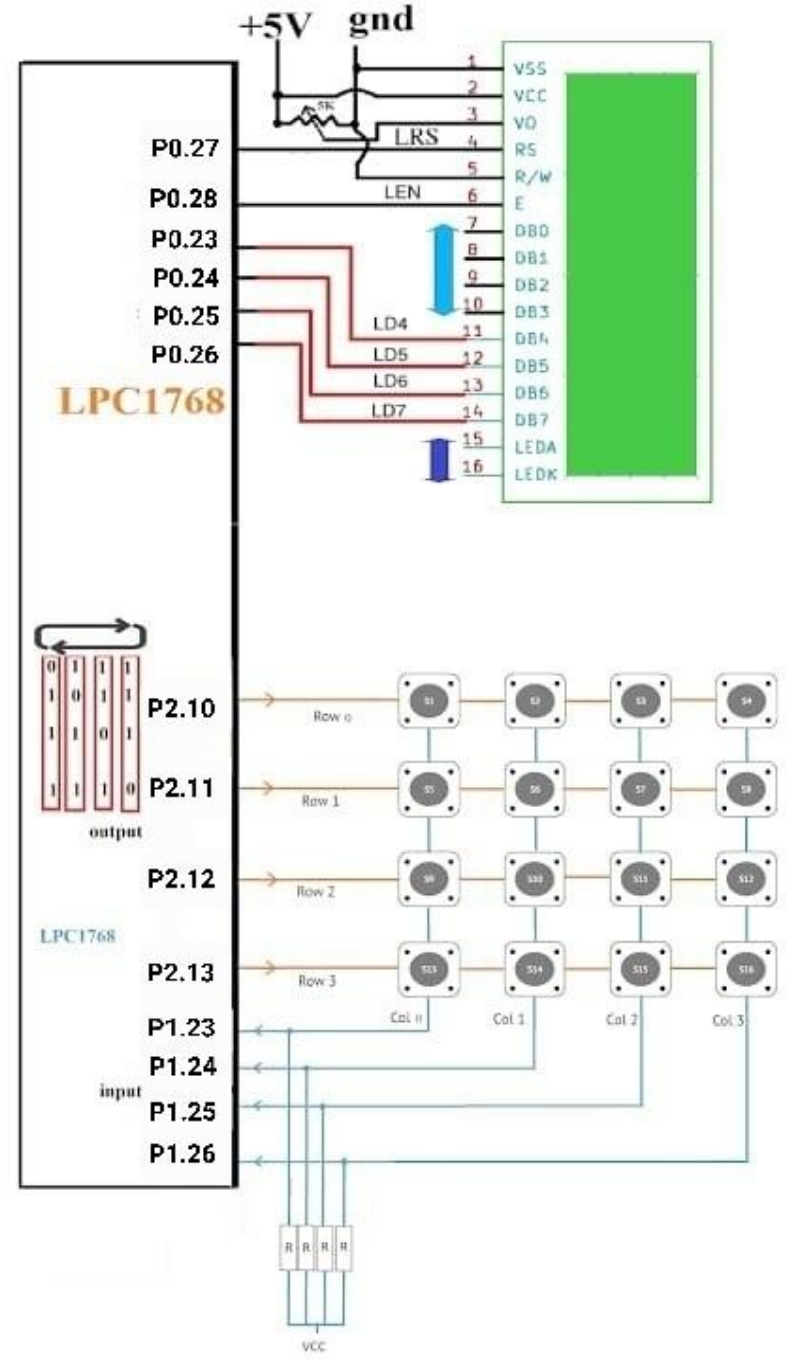
  P0.26 - LD7
  P0.25 - LD6
  P0.24 - LD5
  P0.23 - LD4
  P0.28 - LEN
  P0.27 - LRS

- KEYPAD MATRIX

  C0- P1.23
  C1- P1.24
  C2- P1.25
  C3- P1.26.

  R0- P2.10
  R1- P2.11
  R2- P2.12
  R3- P2.13

CHAPTER 4: BLOCK DIAGRAM



This block diagram shows how to interface a 2x16 LCD with LPC1768 in 4-bit mode. As per the name the 2x16 has 2 lines with 16 chars on each line. It supports all the ascii chars and is basically used for displaying the alphanumeric characters. Here each character is displayed in a matrix of 5x7 pixels.

CHAPTER 5: PROGRAM

LCD.c :

```
#include<LPC17xx.h>

void clear_ports()
{
	LPC_GPIO0->FIOCLR =0xF<<23;
	LPC_GPIO0->FIOCLR =1<<27;
	LPC_GPIO0->FIOCLR =1<<28;
}

void delay_lcd(unsigned int r)
{
	unsigned int t;
	for(t=0;t<r;t++);
}

void write(int temp2,int type)
{
	clear_ports();
	LPC_GPIO0->FIOPIN=temp2;
	if(!type)
	{
		LPC_GPIO0->FIOCLR = 1<<27;
	}
	else
	{
		LPC_GPIO0->FIOSET = 1<<27;
	}
	LPC_GPIO0->FIOSET = 1<<28;
	delay_lcd(25);
	LPC_GPIO0->FIOCLR = 1<<28;
	return;
}
```

```
void lcd_comdata(int temp1,int type)
{
	int temp2 = temp1&0xF0;
	temp2 <<= 19;
	write(temp2,type);
	temp2=temp1&0x0F;
	temp2 <<= 23;
	write(temp2,type);
	delay_lcd(1000);
	return ;
}

void lcd_init()
{
	LPC_GPIO0->FIODIR |= 0xF<<23|1<<27|1<<28;
	clear_ports();
	delay_lcd(3200);

	lcd_comdata(0x33,0);
	delay_lcd(30000);

	lcd_comdata(0x32,0);
	delay_lcd(30000);

	lcd_comdata(0x28, 0);
	delay_lcd(30000);

	lcd_comdata(0x0c, 0);
	delay_lcd(800);

	lcd_comdata(0x06, 0);
	delay_lcd(800);

	lcd_comdata(0x01, 0);
	delay_lcd(10000);

	return;
}
```

```
void lcd_puts(unsigned char* str)
{
	unsigned int temp3,i=0;
	while(str[i])
	{
		temp3=str[i];
		lcd_comdata(temp3,1);
		i++;
		if(i==16)
		{
			lcd_comdata(0xC0,0);
		}
	}
	return;
}
```

MAIN.c :

```
#include <LPC17xx.h>
#include "lcd.c"  // Make sure to import the corresponding file for your LCD functions
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

void scan(void);
unsigned char row,flag,key;
unsigned long int i,j,var1,temp,temp2,temp3;
unsigned char scan_code[16]={0x11,0x21,0x41,0x81,0x12,0x22,0x42,0x82,0x14,0x24,0x44,0x84,0x18,0x28,0x48,0x88};
unsigned char ascci_code[16]={'1','2','3','@','4','5','6','%','7','8','9','&','*','0','#','_'};
unsigned char KEY;
unsigned int counter;
int index=0;
unsigned char Msg3={"Memory full"};
unsigned char Msg4={"Memory empty"};
```

```
unsigned char output[32];
unsigned char Temp_output[128];
unsigned char keypad_chars[8];
unsigned char dvalue;
int nkey = 0;
int npresses = 0;
int nout_idx = 0;
int num_chars = 0;
#define MAX_SIZE 128
typedef struct {
   int front;
   int rear;
   unsigned char data[MAX_SIZE];
} Queue;

Queue queue;
queue.front = queue.rear = -1;

// Mapping of numbers to their corresponding letters on a Nokia keypad
unsigned char* nokia_keypad[] = {
   " 0", ".,?1", "ABC2", "DEF3",
   "GHI4", "JKL5", "MNO6",
   "PQRS7", "TUV8", "WXYZ9"
};


int is_full(Queue queue) {
   if(queue.front == queue.rear){
      lcd_puts(Msg3);
      delay_lcd(800);
      return 1;
   }
   return 0;
}


int is_empty(Queue queue) {
   if(queue.front == -1){
```

```
        lcd_puts(Msg4);
        delay_lcd(800);
        return 1;
    }
    return 0;
}

void enqueue(Queue queue, unsigned char value) {
    if (is_full(queue)) {
        printf("Error: Queue is full!\n");
        return;
    }
    if (is_empty(queue)) {
        queue.front = 0;
    }
    queue.data[queue.rear++] = value;
}


char dequeue(Queue queue) {
    if (is_empty(queue)) {
        printf("Error: Queue is empty!\n");
        return -1;
    }
    dvalue = queue.data[queue.front];
    if (queue.front == queue.rear) {
        queue.front = queue.rear = -1; // Reset the queue
    } else {
        queue.front = queue.front + 1;
    }
    return dvalue;
}

void queue_to_array(Queue q,unsigned char arr[]) {
    index = 0;
    while (!is_empty(q)) {
        arr[index++] = dequeue(q);
    }
```

```
    arr[index] = '\0'; // Null-terminate the character array
}

void print_on_LCD(unsigned char arr[]){
    lcd_comdata(0x01,0);
    delay_lcd(800);
    lcd_puts(&arr[0]);
}

char get_char_from_keypad(int key, int presses) {
    keypad_chars = nokia_keypad[key];
    num_chars = 0;
    for(j=0;keypad_chars[j]!='\0';j++){
        num_chars++;
    }
    return keypad_chars[(presses - 1) % num_chars];
}

void decode_nokia_keypad(unsigned char input[], unsigned char output[]) {
    nkey = 0;
    npresses = 0;
    nout_idx = 0;
    for (i = 0; input[i] != '\0'; i++) {
        if (input[i] >= '0' && input[i] <= '9') {
            nkey = input[i] - '0';
            npresses++;
        } else if (input[i] == '_') {
            output[nout_idx++] = get_char_from_keypad(nkey, npresses);
            presses = 0;
        }
    }
    if (presses > 0) {
        output[nout_idx++] = get_char_from_keypad(nkey, npresses);
    }
    output[nout_idx] = '\0';
}

int main(void)
```

```
{
   SystemInit();
   SystemCoreClockUpdate();
   //Initialize LCD
   lcd_init();
   lcd_comdata(0x80,0);
   delay_lcd(800);


   //Initialization for keyboard
    LPC_GPIO2.FIODIR |= 0x00003C00; // Port 2.10 - 2.13 are made output
ports
    LPC_GPIO1.FIODIR &= 0xF87FFFFF; // Port 1.23- 1.26 are made input
ports
   counter = 1;

   while(1){

   while (1)
   {
      while (1)
      {
         for (row = 1; row < 5; row++)
         {
            if (row == 1)
               var1 = 0x00000400;
            else if (row == 2)
               var1 = 0x00000800;
            else if (row == 3)
               var1 = 0x00001000;
            else if (row == 4)
               var1 = 0x00002000;

            temp = var1;
              LPC_GPIO2.FIOCLR = 0x00003C00; // Clear the ports and send
appropriate data
            LPC_GPIO2.FIOSET = var1;       // Enabling rows
            flag = 0;
```

```
            scan();
            if (flag == 1)
                break;
        } // End for for loop
        if (flag == 1)
            break;
    } // End 2nd While Loop

    for (i = 0; i < 16; i++)
    {
        if (key == scan_code[i])
        {
            KEY = ascci_code[i];
             if(KEY != '#'){ // # is considered to be the termination character of
the string
                enqueue(queue, KEY);
            }
            break;
        }
    }
    if(KEY == '#'){
        break;
    }
  }

  queue_to_array(queue,Temp_output);
  decode_nokia_keypad(Temp_output, output);
  print_on_LCD(output);
  }
}

void scan()
{
  temp3 = LPC_GPIO1.FIOPIN;
  temp3 &= 0x07800000;
  if (temp3 != 0x00000000)
  {
    flag = 1;
```

```
        temp3 >>= 19;
        temp >>= 10;
        key = temp3 | temp; // To get scan code
    } // if temp3!= 0x00000000
}
```