14.Keypads

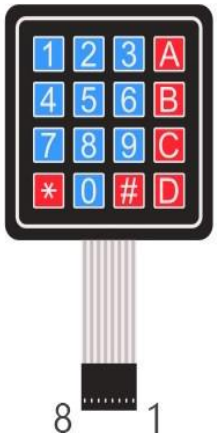
Introduction

In this lesson, we will use a keypad and a LCD to make a combination lock. The LCD will display a corresponding prompt for you to type your password on the Keypad. If the password is input correctly, “Correct” will be displayed.

On the basis of this project, we can add additional electronic components, such as buzzer, LED and so on, to add different experimental phenomena for password input.

Hardware Required

- ✓ 1 * Raspberry Pi
- ✓ 1 * T-Extension Board
- ✓ 1 * 40-pin Cable
- ✓ Several Jumper Wires
- ✓ 1 * Breadboard
- ✓ 1 * I2C LCD1602
- ✓ 1 * Keypads
- ✓ 8 * Resistor 10KΩ

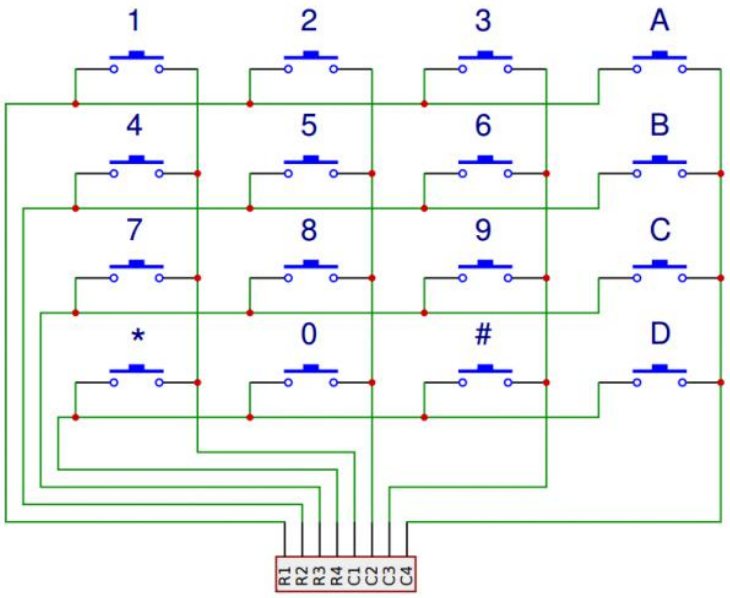


Principle

Keypads

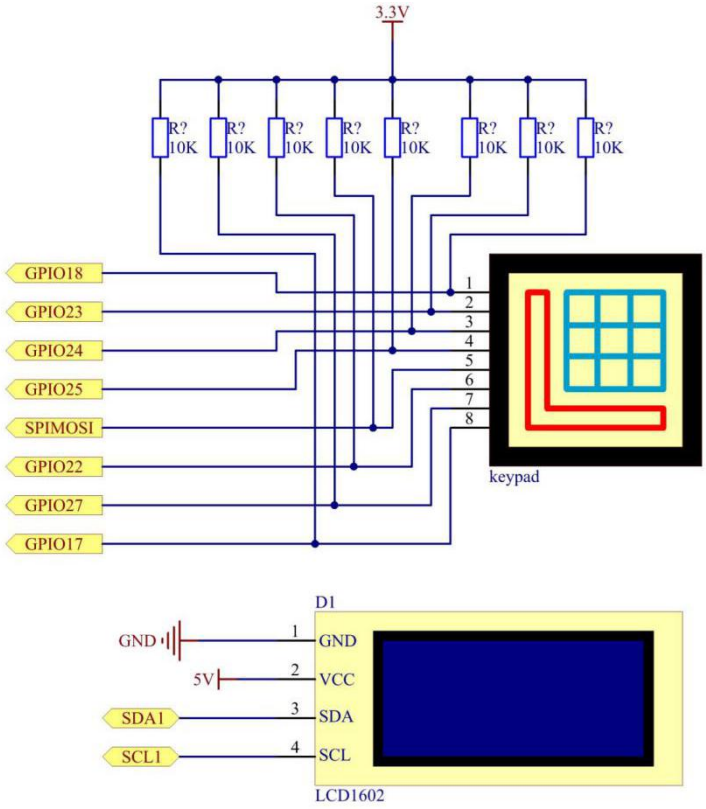
Membrane Switch Module（Keypads）are a great way to let users interact with your project. You can use them to navigate menus, enter passwords, and control games and robots.Pressing a button closes the switch between a column and a row trace, allowing current to flow between a column pin and a row pin.

The schematic for a 4X4 keypad shows how the rows and columns are connected:



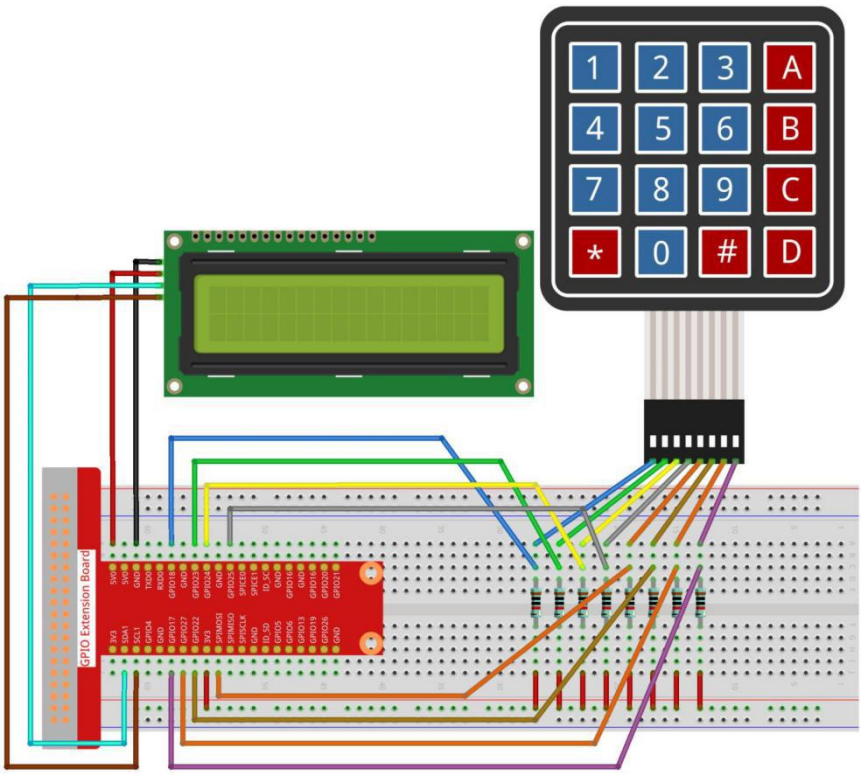
Schematic Diagram

T-Board Name	physical	wiringPi	BCM
GPIO18	Pin 12	1	18
GPIO23	Pin 16	4	23
GPIO24	Pin 18	5	24
GPIO25	Pin 22	6	25
GPIO17	Pin 11	0	17
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22
SPIMOSI	Pin 19	12	10
SDA1	Pin 3		
SCL1	Pin 5		



Experimental Procedures

Step 1: Build the circuit.



For C Language Users

Step 2: Change directory.

```
cd /home/pi/REXQualis_Raspberry_Pi_Complete_Starter_Kit/C/14.Keypads
```

Step 3: Compile.

```
gcc 14.Keypads.cpp Key.cpp Keypad.cpp -o Keypads.out -lwiringPi
```

Note: The program contains custom headers that are compiled when CPP files are compiled.

You can use this method to simplify the instruction.

```
gcc *.cpp -lwiringPi
```

Note: Here we use the wildcard (*), which causes all the files that conform to the format to be processed together.

Step 4: Run the executable file above.

```
sudo ./Keypads.out
```

After the code runs, keypad is used to input password. If the “CORRECT” appears on LCD1602, there is no wrong with the password; otherwise, “WRONG KEY” will appear.

Code

Note: The following codes are incomplete. If you want to check the complete codes, you are suggested to use command nano 14.Keypads.cpp.

```
#include "Keypad.hpp"
#include <stdio.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <string.h>
int LCDAddr = 0x27;
int BLEN = 1;
int fd;
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
```

```
const byte LENS = 4; //password length
char keys[ROWS][COLS] = {    //key code
      {'1','2','3','A'},
      {'4','5','6','B'},
      {'7','8','9','C'},
      {'*','0','#','D'}
};
char password[LENS]={'1','9','8','4'}; //password
char testword[LENS]={};
int keyIndex=0;
byte rowPins[ROWS] = {1, 4, 5, 6 }; //connect to the row pinouts of the keypad
byte colPins[COLS] = {12,3, 2, 0 }; //connect to the column pinouts of the keypad
//create Keypad object
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
void write_word(int data){……}
void send_command(int comm){……}
void send_data(int data){……}
void lcdInit(){……}
void clear(){……}
void write(int x, int y, char const data[]){……}
int check(){
      for(int i=0;i<LENS;i++){
            if(password[i]!=testword[i])
            {return 0;}
      }
      return 1;
}
int main(){
      if(wiringPiSetup() == -1){……}
      fd = wiringPiI2CSetup(LCDAddr);
```

```
lcdInit();
clear();
write(0, 0, "WELCOME!");
write(2, 1, "Enter password");
char key = 0;
keypad.setDebounceTime(50);
while(1){
      key = keypad.getKey();    //get the state of keys
      if (key){           //if a key is pressed, print out its key code
            clear();
            write(0, 0, "Enter password:");
            write(15-keyIndex,1, "****");
            testword[keyIndex]=key;
            keyIndex++;
            if(keyIndex==LENS){
                  if(check()==0){
                        clear();
                        write(3, 0, "WRONG KEY!");
                        write(0, 1, "please try again");
                        }
                  else{
                        clear();
                        write(4, 0, "CORRECT!");
                        write(2, 1, "welcome back");
                        }
            }
            keyIndex=keyIndex%LENS;
      }
}
return 1;
```

```
}
```

Code Explanation

```
const byte LENS = 4; //password length
char password[LENS]={'1','9','8','4'}; //password
char testword[LENS]={};
int keyIndex=0;
```

The variable, password[LENS] is the correct password that we've set in the program; testword[LENS] is used to store characters input during program operation. KeyIndex is used to indicate the number of digits input.

```
int check(){
    for(int i=0;i<LENS;i++){
        if(password[i]!=testword[i])
        {return 0;}
    }
    return 1;
}
```

We use a loop to compare the input result with the default password bit by bit. The true value is returned only if they correspond exactly, otherwise the return value is false.

```
while(1){
  key = keypad.getKey();    //get the state of keys
  if (key){            //if a key is pressed, print out its key code
        clear();
       write(0, 0, "Enter password:");
        write(15-keyIndex,1, "****");
        testword[keyIndex]=key;
        keyIndex++;
        if(keyIndex==LENS){
            if(check()==0){
                clear();
                write(3, 0, "WRONG KEY!");
                write(0, 1, "please try again");
                }
            else{
                clear();
                write(4, 0, "CORRECT!");
                write(2, 1, "welcome back");
                }
        }
        keyIndex=keyIndex%LENS;
```

Get the value of keypad when it is pressed. Then the value input will be displayed on LCD. The variable keyIndex is used to change the display position of "****" according to the password digits input, and since positions greater than 15 are beyond the screen. the effect is to add a "*" on the LCD display for each digit input.

Then assign the input value to the testword[] array and add 1 to keyIndex. When keyIndex reaches the length of the password, the program judges whether the return value of the check() function is 0. If so, it indicates that the password is wrong and an error message will be displayed on LCD1602; otherwise, LCD1602 displays "CORRECT"

and "welcome back".

For Python Language Users

Step 2: Change directory.

```
cd /home/pi/REXQualis_Raspberry_Pi_Complete_Starter_Kit/Python
```

Step 3: Run.

```
sudo python3 14.Keypads.py
```

After the code runs, keypad is used to input password:1984. If the “ CORRECT ”appears on LCD1602, there is no wrong with the password; otherwise, “WRONG KEY”will appear.

Code

Note: The following codes are incomplete. If you want to check the complete codes,you are suggested to use command nano 14.Keypads.py.

```
import RPi.GPIO as GPIO
import time
import LCD1602
################ HERE IS THE KEYPAD################
class Key(object):……
class Keypad(object):……
################EXAMPLE CODE START HERE ################
ROWS = 4
COLS = 4
LENS = 4
keys = [     '1','2','3','A',
             '4','5','6','B',
             '7','8','9','C',
             '*','0','#','D'       ]
password=['1','9','8','4']
testword=['0','0','0','0']
keyIndex=0
```

```
rowsPins = [12,16,18,22]
colsPins = [19,15,13,11]
def check():
    for i in range(0,LENS):
        if(password[i]!=testword[i]):
            return 0
    return 1
def setup():
    LCD1602.init(0x27, 1)       # init(slave address, background light)
    LCD1602.clear()
    LCD1602.write(0, 0, 'WELCOME!')
    LCD1602.write(2, 1, 'Enter password')
    time.sleep(2)
def destroy():
    LCD1602.clear()
def loop():
    keypad = Keypad(keys,rowsPins,colsPins,ROWS,COLS)
    keypad.setDebounceTime(50)
    global keyIndex
    global LENS
    while(True):
        key = keypad.getKey()
        if(key != keypad.NULL):
            LCD1602.clear()
            LCD1602.write(0, 0, "Enter password:")
            LCD1602.write(15-keyIndex,1, "****")
            testword[keyIndex]=key
            keyIndex+=1
            if (keyIndex is LENS):
                if (check() is 0):
```

```
                LCD1602.clear()
                LCD1602.write(3, 0, "WRONG KEY!")
                LCD1602.write(0, 1, "please try again")
            else:
                LCD1602.clear()
                LCD1602.write(4, 0, "CORRECT!")
                LCD1602.write(2, 1, "welcome back")
        keyIndex=keyIndex%LENS
if __name__ == '__main__':      # Program start from here
    try:
        setup()
        loop()
except KeyboardInterrupt:   # When 'Ctrl+C' is pressed, the program destroy() will be executed.
        destroy()
```

Code Explanation

```
LENS = 4
password=['1','9','8','4']
testword=['0','0','0','0']
keyIndex=0
```

The variable, password is the correct password that we've set in the program; testword is used to store characters entered during program operation. KeyIndex is used to indicate the number of digits input.

```
def check():
    for i in range(0,LENS):
        if(password[i]!=testword[i]):
            return 0
    return 1
```

We use a loop to compare the input result with the default password bit by bit. The true value is returned only if they correspond exactly, otherwise the return value is false.

```
        def setup():
    LCD1602.init(0x27, 1)      # init(slave address, background light)
    LCD1602.clear()
    LCD1602.write(0, 0, 'WELCOME!')
    LCD1602.write(2, 1, 'Enter password')
    time.sleep(2)
```

The function setup() is used to desplay I2C LCD1602. LCD1602.init(0x27, 1) is used to initialize the address of LCD1602 and turn on the backlight. Lcd1602.clear() is applied to clear the screen if it is necessary, or the characters will overlap and affect the display result. LCD1602.write(0, 0, 'WELCOME!') is used to display the position of characters. In row 0, column 0, character 'WELCOME' will appear.

Read the key of keypad when it is pressed, then the key input will appear on LCD.

```
        key = keypad.getKey()
        if(key != keypad.NULL):
            LCD1602.clear()
            LCD1602.write(0, 0, "Enter password:")
            LCD1602.write(15-keyIndex,1, "****")
            testword[keyIndex]=key
            keyIndex+=1
            if (keyIndex is LENS):
                if (check() is 0):
                    LCD1602.clear()
                    LCD1602.write(3, 0, "WRONG KEY!")
                    LCD1602.write(0, 1, "please try again")
                else:
                    LCD1602.clear()
                    LCD1602.write(4, 0, "CORRECT!")
                    LCD1602.write(2, 1, "welcome back")
            keyIndex=keyIndex%LENS
```

The variable keyIndex is used to change the display position of "****" according to the password digits input since positions greater than 15 are beyond the screen. The effect is to add a "*" on the LCD display for each digit input.

Then assign the input value to the testword[] array and add 1 to keyIndex. When keyIndex reaches the length of the password, judge whether the return value of check() function is 0; if the condition is met, it means that the password input is wrong; accordingly, the notice of error displays on LCD1602. Otherwise, there appears “CORRECT” and “welcome back”.

Phenomenon Picture