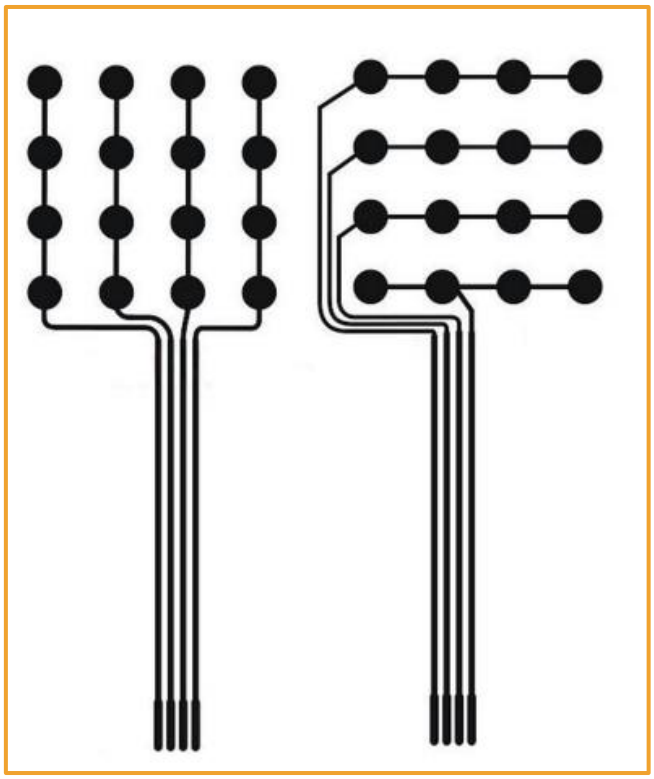
Matrix Keyboard Experiment

Introduction

The matrix keyboard used in this experiment contains 16 keys, 4 rows and 4 columns. The 4 keys in each row are connected together and the 4 keys in each column are also connected together, thus forming a matrix keyboard with 4 rows and 4 columns.

Matrix Keyboard Schematic Diagram

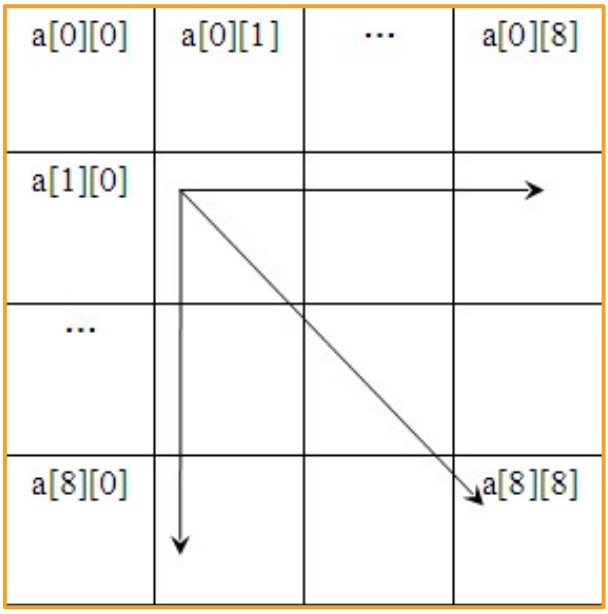


Working Principle

First of , we send a low level voltage to the column and the remaining columns maintain a high level voltage, and then cycle through each row. If the low level voltage is not detected, it means that no keys in the column are pressed, and then we continue to send a low level voltages to the remaining columns in turn and scan them. Once the high level voltage is detected, it can be determined which column of the button is pressed. The Raspberry Pi scans and detects fast enough that you do not have to worry about missing pressed key.

In the program, we define and use a 2-dimensional character array. Arrays are designed to number simple types of data, a [0], a [1], a [2], a [3]... However, this data type is not convenient in some cases. For instance, we want to define an array to record the result of the 1-9 multiplication table. The best way to record the results of the 1-9 multiplication

table is not to define an array of length 81, but to define an array of 9 rows and 9 columns. Therefore, the reference array has two subscripts that is the row and the column. Such arrays are called 2-dimensional arrays and so on.



It is obvious that a two-dimensional array is so convenient for recording key character data. For instance, we need to know the data in the 2nd row and 3rd column of the keyboard, as long as we refer to **hexaKeys** [1] [2] in the defined 2-dimensional array **hexaKeys**. At the beginning of the definition of a 2-dimensional array, we can assign a value to it. Take the above 4 x 4 keyboard as an example and the format is::

```
char hexaKeys[ROWS][COLS] = {
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};
```

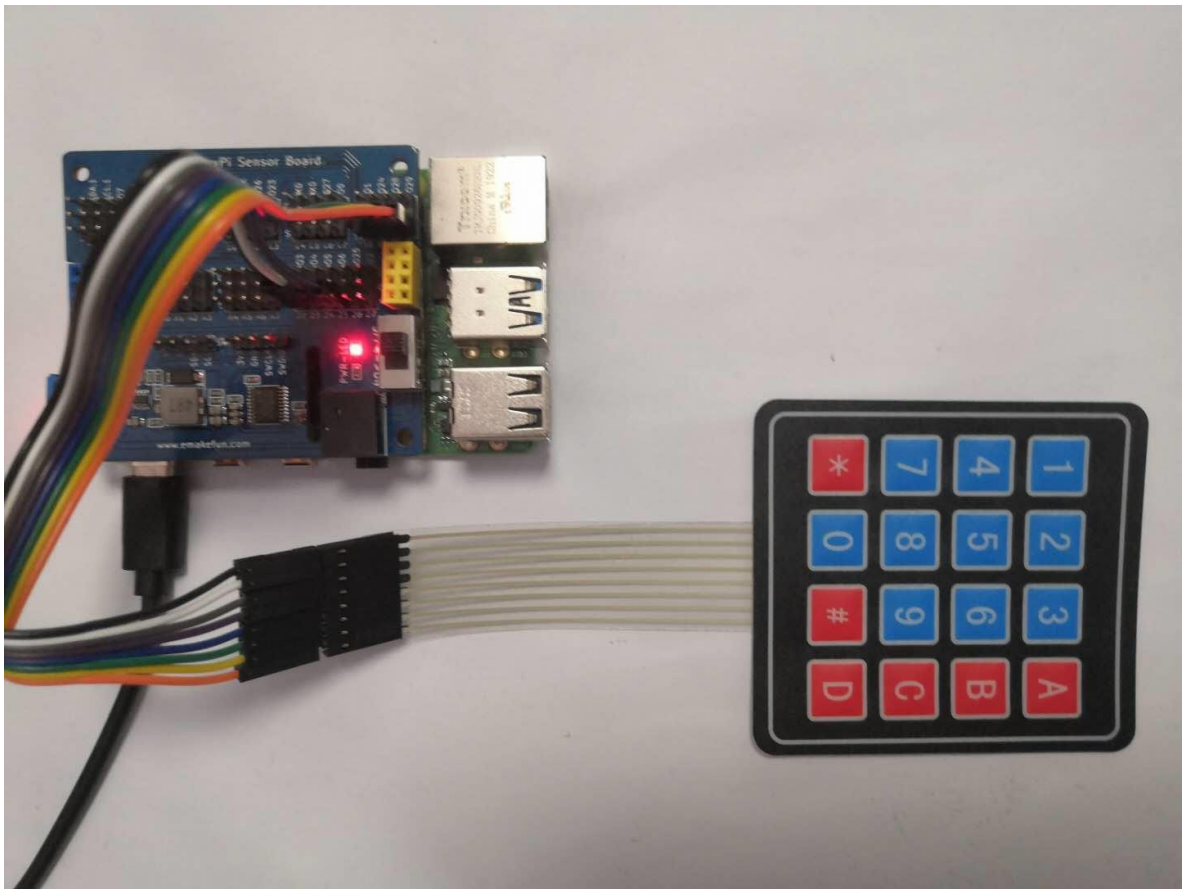
Component List

- Raspberry Pi main board
- Breadboard
- Cable

- Matrix Keyboard * 1
- Several jumper wires

Wiring

Raspberry Pi	Matrix Keyboard
IO3(wiringPi)/22(BCM)	1
IO4(wiringPi)/23(BCM)	2
IO5(wiringPi)/24(BCM)	3
IO6(wiringPi)/25(BCM)	4
IO1(wiringPi)/18(BCM)	5
IO24(wiringPi)/19(BCM)	6
IO28(wiringPi)/20(BCM)	7
IO29(wiringPi)/21(BCM)	8



C++ partial program

```
#include "key4x4.h"

uint8_t pin_val[8] = {3, 4, 5, 6, 1, 24, 28, 29};
char key[4][4] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

void getval()
```

```
{
        for (int i=0; i<4; i++)
        {
            out_set(i);
            for (int j=0; j<4; j++)
            {
                if (digitalRead(pin_val[j+4]))
                {
                    delay(100);
                    if (digitalRead(pin_val[j+4]))
                    {
                        printf("key: %c\n",key[i][j]);
                    }
                }
            }
        }


}


void out_set(int n)
{
    pinMode(pin_val[0],OUTPUT);
    pinMode(pin_val[1],OUTPUT);
    pinMode(pin_val[2],OUTPUT);
    pinMode(pin_val[3],OUTPUT);
    pinMode(pin_val[4],INPUT);
    pinMode(pin_val[5],INPUT);
    pinMode(pin_val[6],INPUT);
    pinMode(pin_val[7],INPUT);
    switch(n){
            case 0:
                digitalWrite(pin_val[0],HIGH);
                digitalWrite(pin_val[1],LOW);
                digitalWrite(pin_val[2],LOW);
                digitalWrite(pin_val[3],LOW);
                break;
            case 1:
                digitalWrite(pin_val[0],LOW);
                digitalWrite(pin_val[1],HIGH);
                digitalWrite(pin_val[2],LOW);
```

```
            digitalWrite(pin_val[3],LOW);
            break;
        case 2:
            digitalWrite(pin_val[0],LOW);
            digitalWrite(pin_val[1],LOW);
            digitalWrite(pin_val[2],HIGH);
            digitalWrite(pin_val[3],LOW);
            break;
        case 3:
            digitalWrite(pin_val[0],LOW);
            digitalWrite(pin_val[1],LOW);
            digitalWrite(pin_val[2],LOW);
            digitalWrite(pin_val[3],HIGH);
            break;
        default:
            break;
    }
}
```

Python program

```
import RPi.GPIO as GPIO
import time

pin_val = [22, 23, 24, 25, 18, 19, 20, 21]
key = [
  ['1','2','3','A'],
  ['4','5','6','B'],
  ['7','8','9','C'],
  ['*','0','#','D']
]

GPIO.setmode(GPIO.BCM)

def out_set(n):
    for i in range(0,4):
        GPIO.setup(pin_val[i],GPIO.OUT)
        GPIO.setup(pin_val[i+4], GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
    if n == 0:
```

```
        GPIO.output(pin_val[0],GPIO.HIGH)
        GPIO.output(pin_val[1],GPIO.LOW)
        GPIO.output(pin_val[2],GPIO.LOW)
        GPIO.output(pin_val[3],GPIO.LOW)
    if n == 1:
        GPIO.output(pin_val[0],GPIO.LOW)
        GPIO.output(pin_val[1],GPIO.HIGH)
        GPIO.output(pin_val[2],GPIO.LOW)
        GPIO.output(pin_val[3],GPIO.LOW)
    if n == 2:
        GPIO.output(pin_val[0],GPIO.LOW)
        GPIO.output(pin_val[1],GPIO.LOW)
        GPIO.output(pin_val[2],GPIO.HIGH)
        GPIO.output(pin_val[3],GPIO.LOW)
    if n == 3:
        GPIO.output(pin_val[0],GPIO.LOW)
        GPIO.output(pin_val[1],GPIO.LOW)
        GPIO.output(pin_val[2],GPIO.LOW)
        GPIO.output(pin_val[3],GPIO.HIGH)

while True:
    for n in range(0,4):
        out_set(n)
        for m in range(0,4):
            if GPIO.input(pin_val[m+4]):
                time.sleep(0.01)
                if GPIO.input(pin_val[m+4]):
                    print(key[n][m])
```

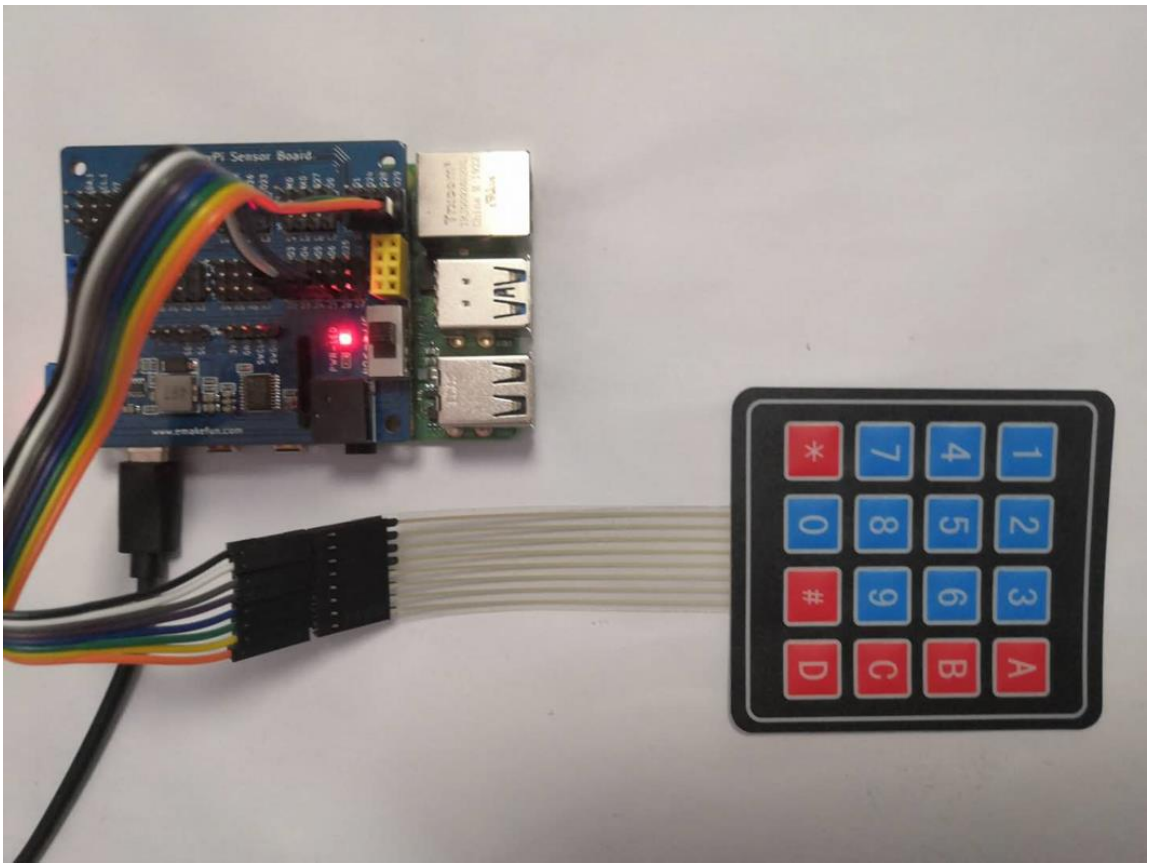
Java program

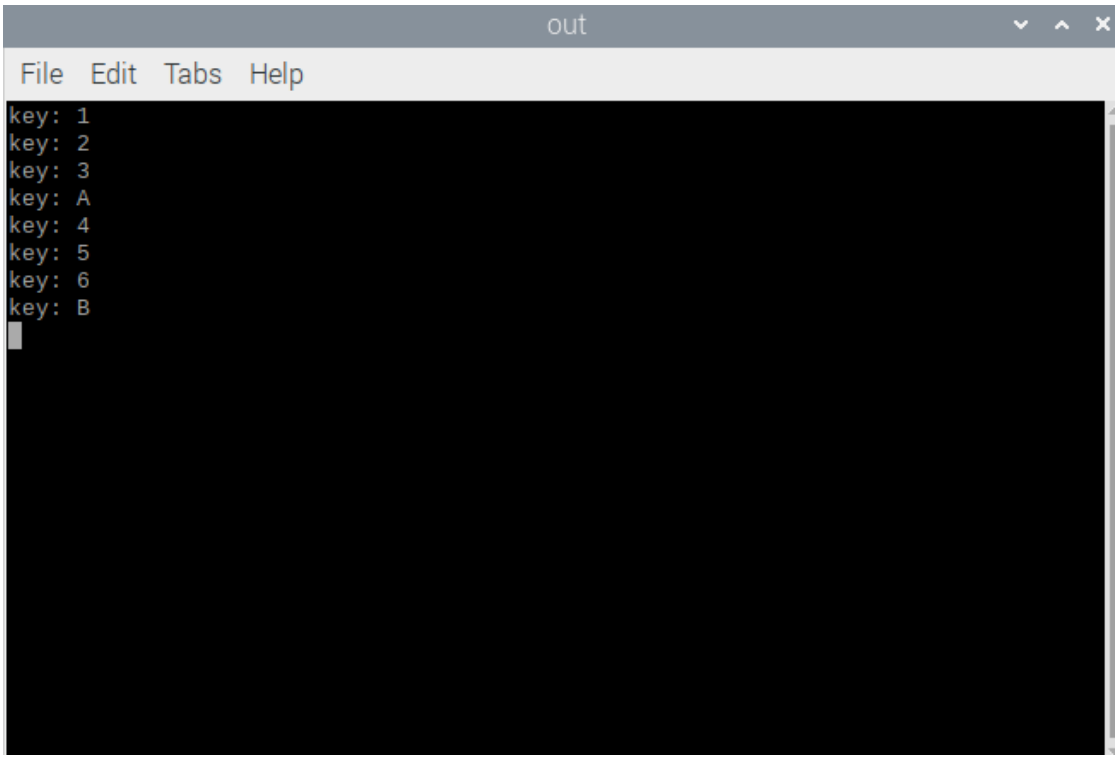
```
import com.pi4j.wiringpi.Gpio;
import com.pi4j.wiringpi.GpioInterrupt;
import com.pi4j.wiringpi.GpioInterruptListener;
import com.pi4j.wiringpi.GpioInterruptEvent;
import com.pi4j.wiringpi.GpioUtil;

public class Matrix_keyboard {
    static char [] pin_val = {3, 4, 5, 6, 1, 24, 28, 29};
    static char key[][] = new char [][] {{'1','2','3','A'}, {'4','5','6','B'},
{'7','8','9','C'}, {'*','0','#','D'}};

    public static void getval () {
```

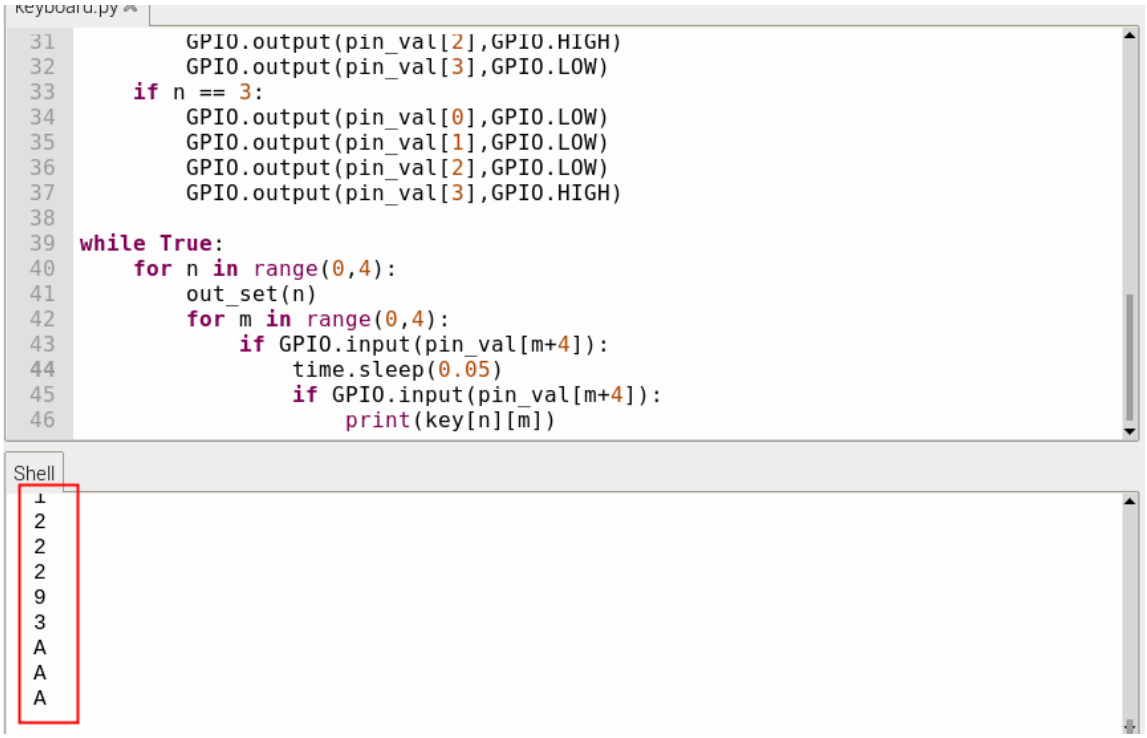
```
    for (int i=0; i<4; i++)
    {
        Matrix_keyboard.out_set(i);
        for (int j=0; j<4; j++)
        {
            if (Gpio.digitalRead(pin_val[j+4]) == 1)
            {
               Gpio.delay(200);
                if (Gpio.digitalRead(pin_val[j+4]) == 1)
                {
                   System.out.println(key[i][j]);
                }
            }
        }
    }
}

public static void out_set (int n) {
    Gpio.pinMode(pin_val[0],Gpio.OUTPUT);
    Gpio.pinMode(pin_val[1],Gpio.OUTPUT);
    Gpio.pinMode(pin_val[2],Gpio.OUTPUT);
    Gpio.pinMode(pin_val[3],Gpio.OUTPUT);
    Gpio.pinMode(pin_val[4],Gpio.INPUT);
    Gpio.pinMode(pin_val[5],Gpio.INPUT);
    Gpio.pinMode(pin_val[6],Gpio.INPUT);
    Gpio.pinMode(pin_val[7],Gpio.INPUT);
    switch(n){
           case 0:
               Gpio.digitalWrite(pin_val[0],Gpio.HIGH);
               Gpio.digitalWrite(pin_val[1],Gpio.LOW);
               Gpio.digitalWrite(pin_val[2],Gpio.LOW);
               Gpio.digitalWrite(pin_val[3],Gpio.LOW);
              break;
           case 1:
               Gpio.digitalWrite(pin_val[0],Gpio.LOW);
               Gpio.digitalWrite(pin_val[1],Gpio.HIGH);
               Gpio.digitalWrite(pin_val[2],Gpio.LOW);
               Gpio.digitalWrite(pin_val[3],Gpio.LOW);
              break;
           case 2:
               Gpio.digitalWrite(pin_val[0],Gpio.LOW);
               Gpio.digitalWrite(pin_val[1],Gpio.LOW);
```

```
                Gpio.digitalWrite(pin_val[2],Gpio.HIGH);
                Gpio.digitalWrite(pin_val[3],Gpio.LOW);
                break;
            case 3:
                Gpio.digitalWrite(pin_val[0],Gpio.LOW);
                Gpio.digitalWrite(pin_val[1],Gpio.LOW);
                Gpio.digitalWrite(pin_val[2],Gpio.LOW);
                Gpio.digitalWrite(pin_val[3],Gpio.HIGH);
                break;
            default:
                break;
        }
    }
    public static void main(String args[]) throws InterruptedException {

        // setup wiring pi
        if (Gpio.wiringPiSetup() == -1) {
            System.out.println(" ==>> GPIO SETUP FAILED");
            return;
        }

        for (;;) {
            Matrix_keyboard.getval();
            //Thread.sleep(1000);
        }
    }
}
```

Experimental Effect







The effect of this experiment is that after we press the button on the matrix keyboard and the value of the button will be displayed on the computer terminal.