

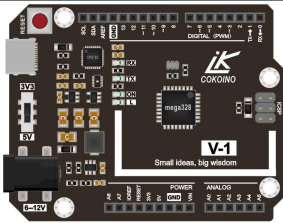
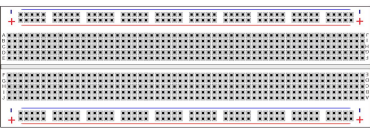

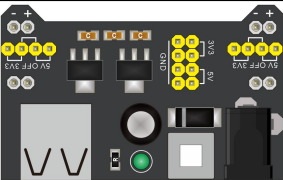



lesson_13 Music keyboard





ABOUT THIS PROJECT:

You will learn:

◆ How to use the 4x4 keyboard and how the 4x4 keyboard works.

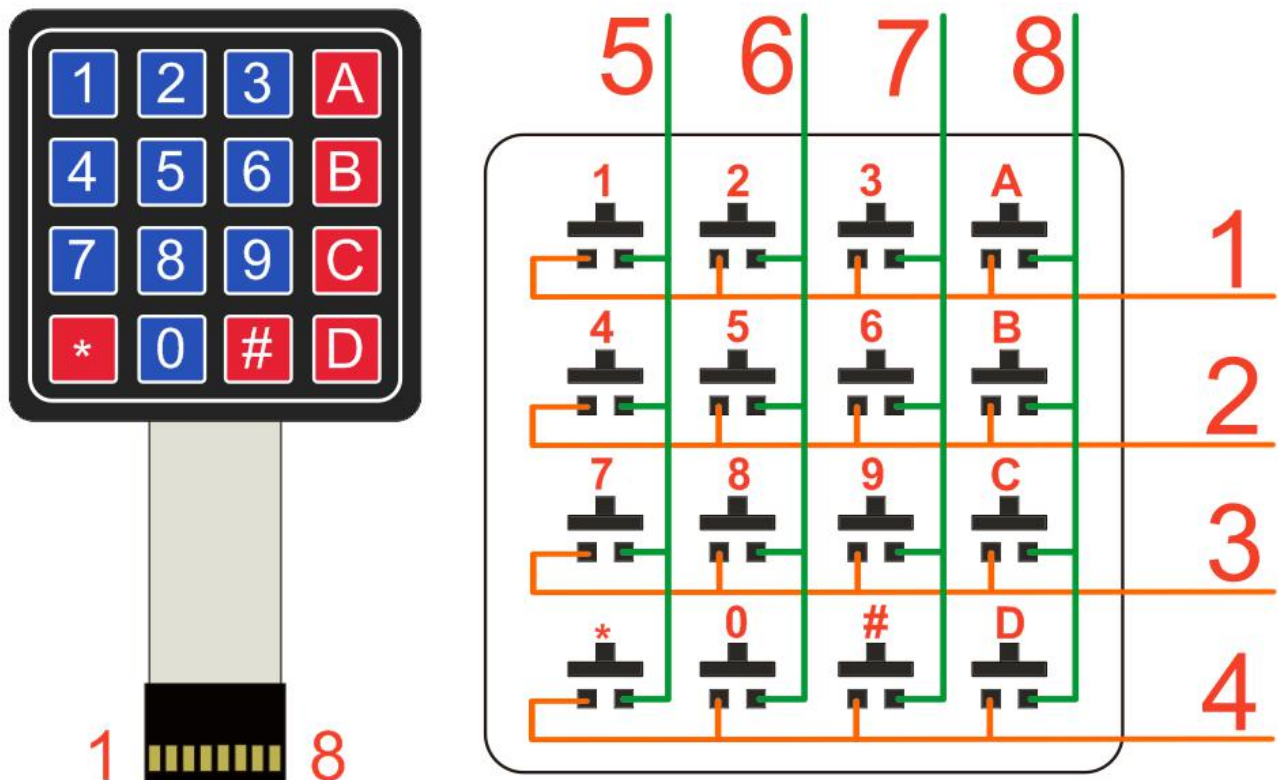
1、 Things used in this project:

Hardware components	Picture	Quantity
V-1 board		1 PCS
Breadboard		1 PCS
Battery button (you need to buy 9V battery yourself)		1 PCS
Breadboard power module		1 PCS
Male to Male DuPont Cable		12 PCS
Type C USB Cable		1 PCS
SS8050 Transistor		1 PCS

IN4148 diode		1 PCS
Active buzzer		1 PCS
4*4 membrane button		1 PCS
220R Resistance		1 PCS

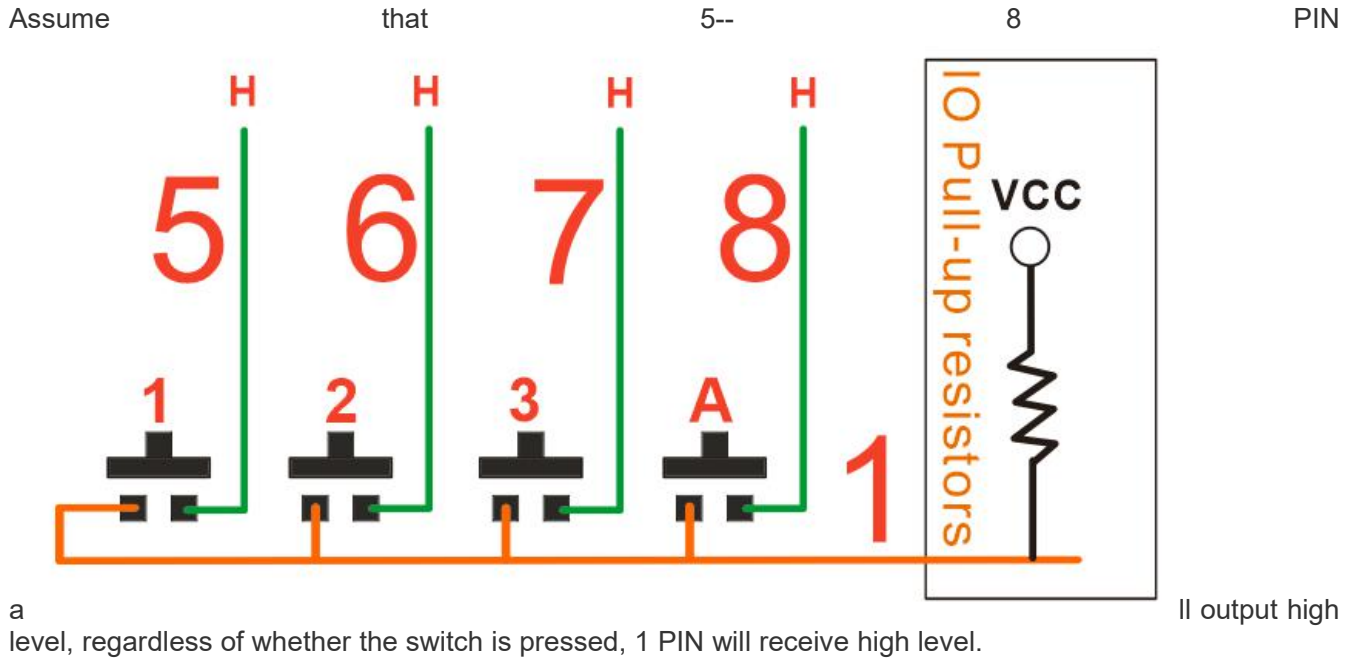
2、 Know the 4 x 4 keyboard.

The 4×4 keyboard has 8 pins, which are divided into columns and rows, as shown in the figure below:

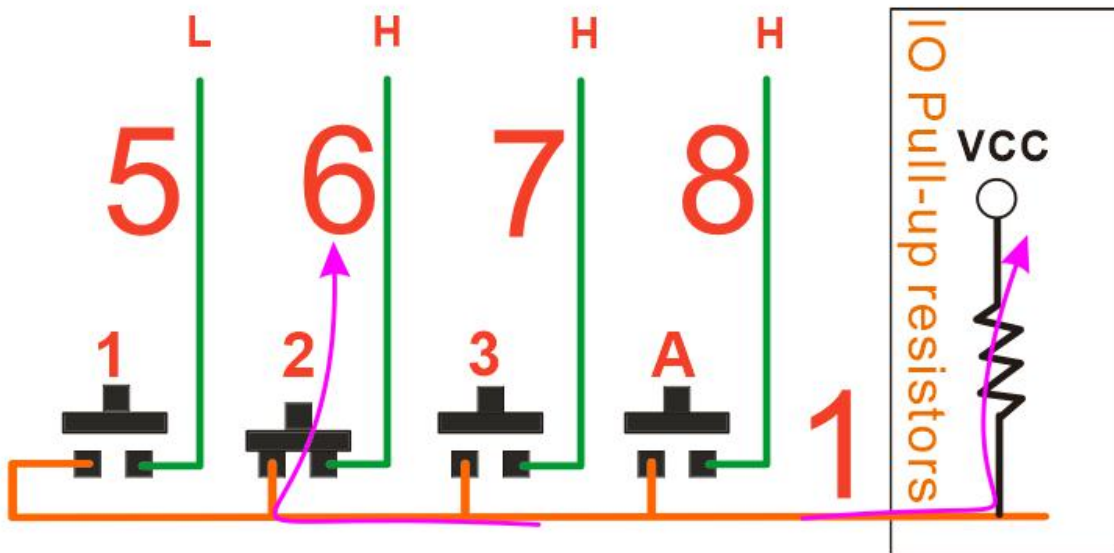


2.1、 Key detection and scanning principle:

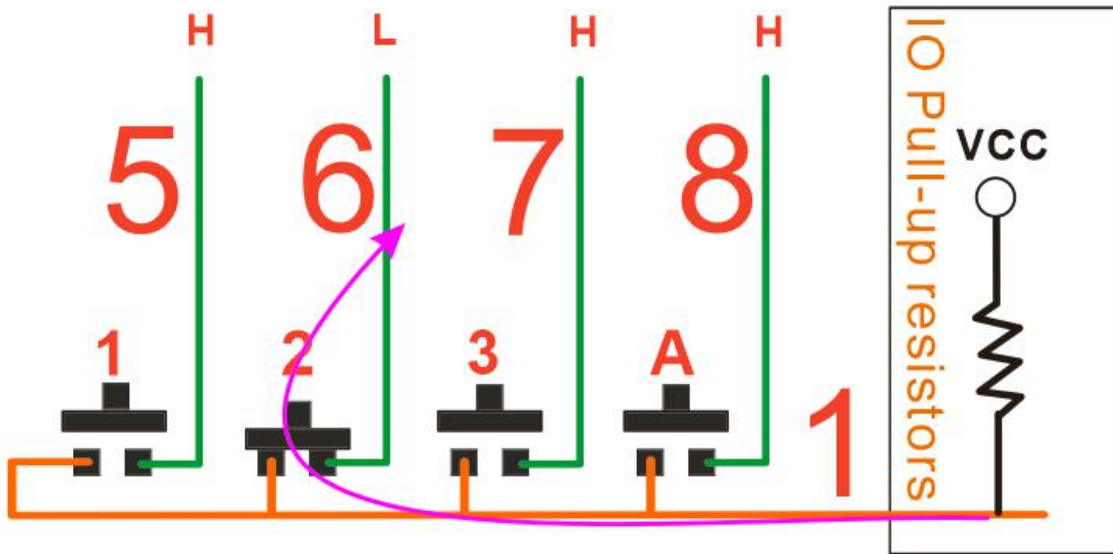
For the sake of explanation, simplify the 4×4 button to 4×1 , as shown below, and connect it to 4 switches. Then, pin 1, 5, 6, 7 and 8 are connected to 5 IO ports of the microcontroller respectively. Pin 5, 6, 7 and 8 are set as output mode, and pin 1 is set as input mode. In addition, to simplify the switching circuit, IO pull-up resistance inside the microcontroller corresponding to pin 1 should be enabled:



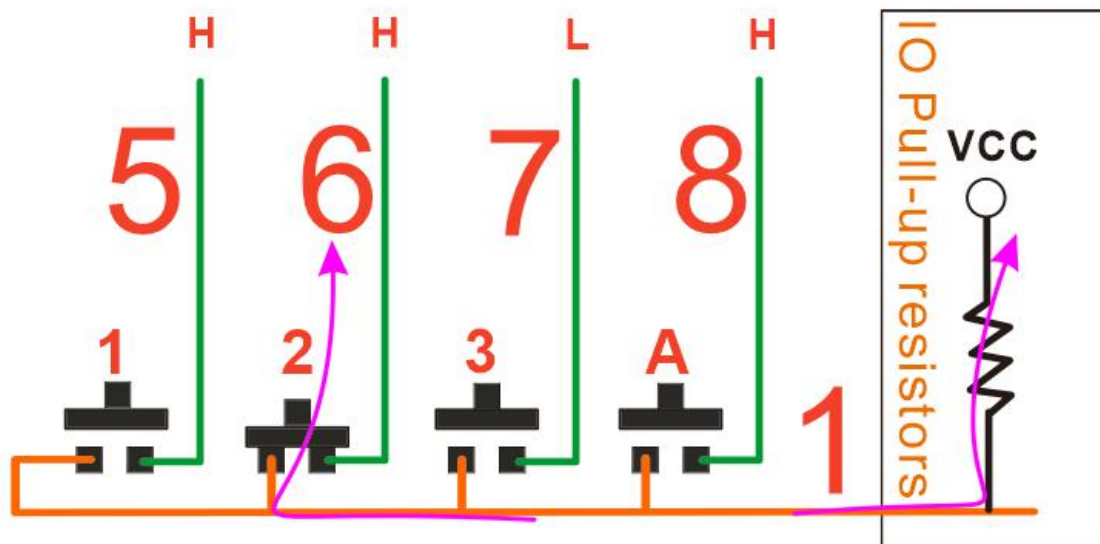
For example, when key 2 is pressed, the program must sequentially set pin5-8 to low level in order to detect that key 2 is pressed. As shown in the figure below, when pin 5 outputs low level and the key does not switch on, pin 1 is still detected high level due to internal pull-up resistance and pin 6 high level, indicating that the key 1 is not pressed.



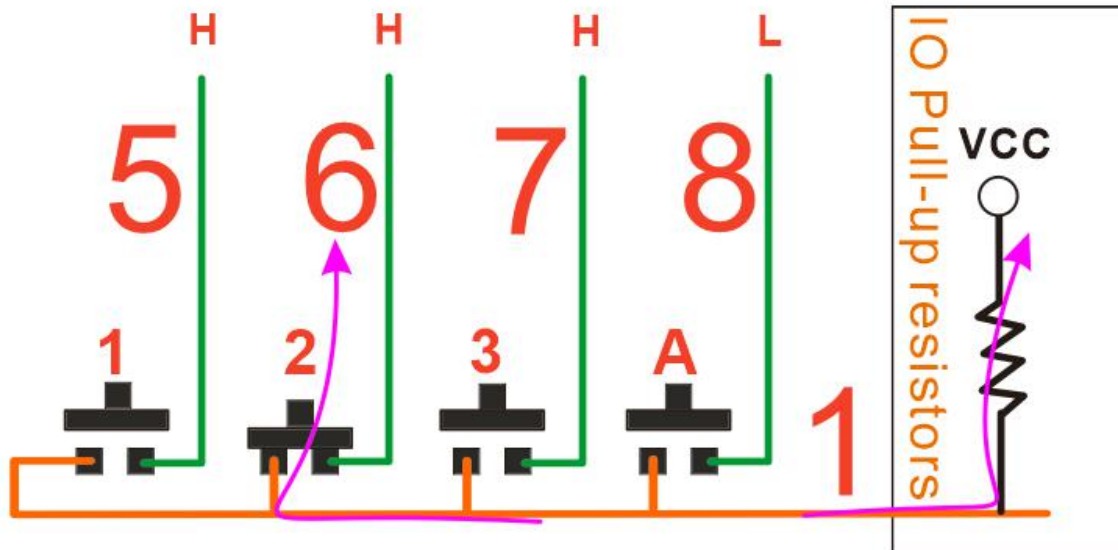
Next, pin6 is input to low level. At this time, pin1 of the microcontroller will receive low level by pressing the button. Therefore, switch 2 connected to Pin6 is pressed.



Then enter the low level at pin 7. Because key 3 was not pressed, and key 3 was not on, pin1 was still detected due to the internal pull-up resistance and high level of Pin6. Therefore, pin1 of the microcontroller received a high level, indicating that key 3 was not pressed.

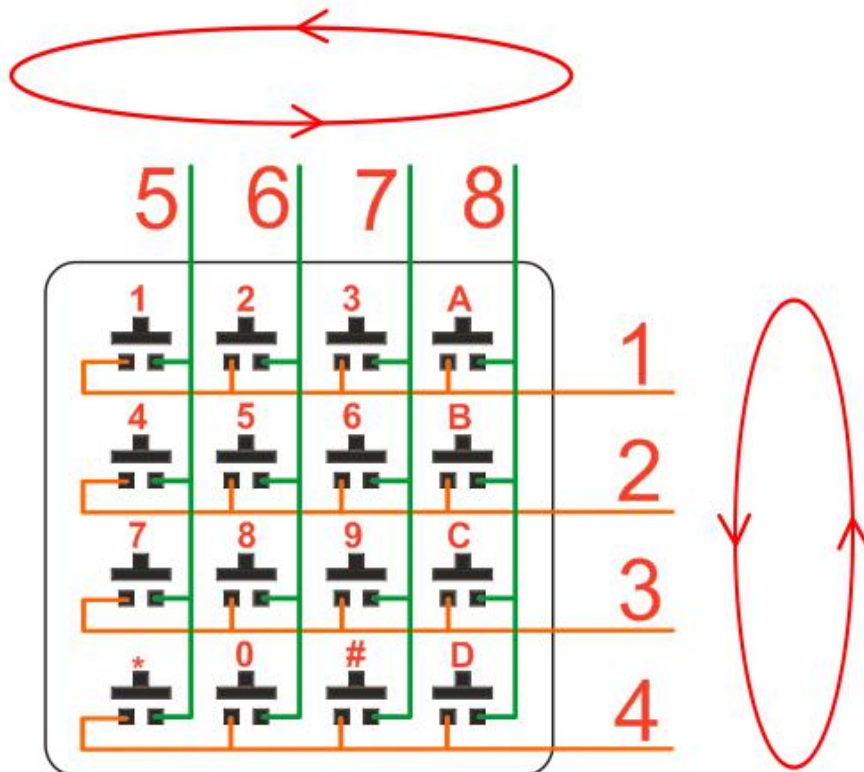


Then enter low level at 8 pins. Because key A was not pressed, the key did not switch on. Pin1 detected high level due to internal pull-up resistance and IO 6 high level, IO 1 of the microcontroller received high level, indicating that key A was not pressed.



At this point, the program that detects the key has to go back to 5 feet again, output low power, and scan repeatedly to detect whether a key has been pressed.

The actual program of 4*4 keyboard press needs to use double loop to scan all keys: the inner loop is PIN5 --8 pins to output low level in sequence; The outer loop is 1-- 4pins detects which switch is pressed in each row in sequence.

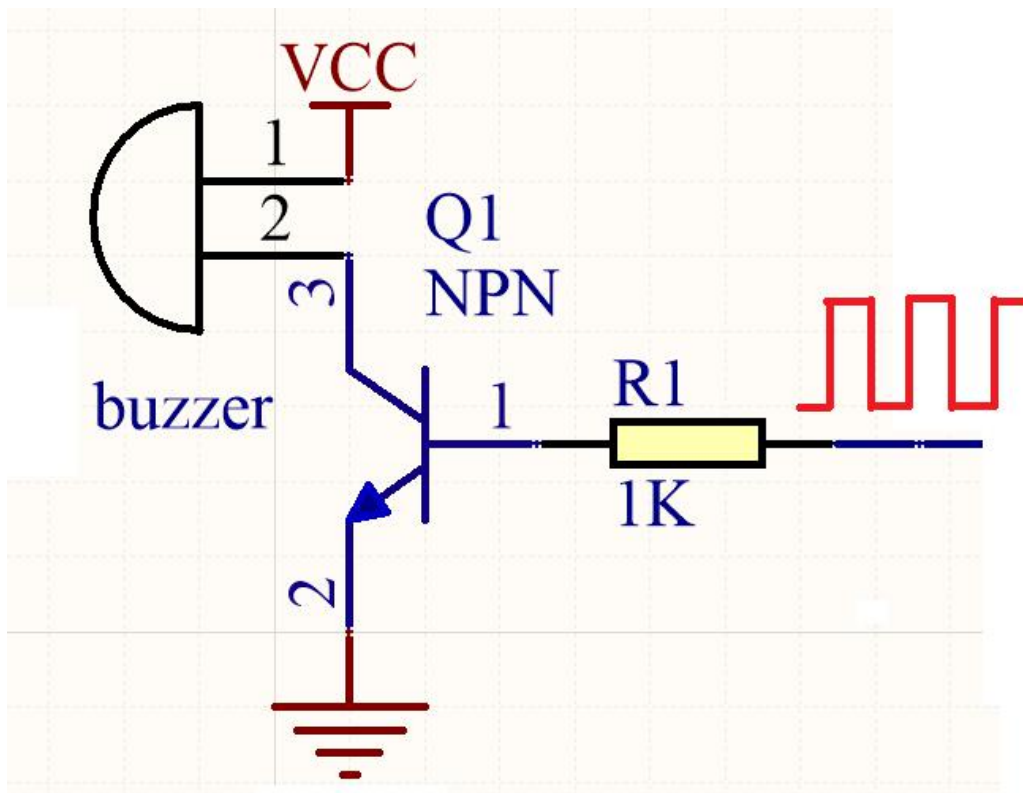


3、 Learn about passive buzzers

There is no oscillation source inside the passive buzzer, which cannot be made to hum with DC voltage. It must be driven by square wave of 2K~5K. The working voltage of this passive buzzer is 5V, the center frequency is 2KHz, and the impedance is 16 Ohms. Generally, the "+" end is connected to the positive pole and the other end to the negative pole, as shown in the figure below:



3.1、 The simple drive circuit is as follows:



4、Music keyboard experiment:

In this experiment, keypad library function is used to read the key value of 4*4 keypad to reduce the difficulty of writing code, so please copy the Keypad library files provided in the course folder to the Libraries folder under the IDE installation directory. The read key values are then converted to PWM signals so that the microphone emits sound of different frequencies and sizes.

The main program statements are: PWM_data = analogRead(A0); AnalogWrite (pin, PWM_data); Adjustment of PMW signal frequency and use of IDE serial port monitor.

Please refer to the following webpage for more PWM and FM information:

<https://playground.arduino.cc/Main/TimerPWMCheatsheet/>

<https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM>

4.1、code

```
#include <Keypad.h>
#define buzzer 11
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the symbols on the buttons of the keypads
char hexaKeys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {2, 3, 4, 5}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {6, 7, 8, 9}; //connect to the column pinouts of the keypad
//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
void setup() {
  Serial.begin(9600);
}
void loop() {
  char customKey = customKeypad.getKey();
  if (customKey) {
    switch(customKey) {
      case '0': setPwmFrequency(11, 8); analogWrite(buzzer, 10); break;
      case '1': setPwmFrequency(11, 8); analogWrite(buzzer, 20); break;
      case '2': setPwmFrequency(11, 8); analogWrite(buzzer, 30); break;
      case '3': setPwmFrequency(11, 32); analogWrite(buzzer, 40); break;
      case '4': setPwmFrequency(11, 32); analogWrite(buzzer, 50); break;
      case '5': setPwmFrequency(11, 32); analogWrite(buzzer, 60); break;
      case '6': setPwmFrequency(11, 64); analogWrite(buzzer, 70); break;
      case '7': setPwmFrequency(11, 64); analogWrite(buzzer, 80); break;
      case '8': setPwmFrequency(11, 64); analogWrite(buzzer, 90); break;
      case '9': setPwmFrequency(11, 128); analogWrite(buzzer, 100); break;
      case 'A': setPwmFrequency(11, 128); analogWrite(buzzer, 110); break;
      case 'B': setPwmFrequency(11, 128); analogWrite(buzzer, 120); break;
      case 'C': setPwmFrequency(11, 256); analogWrite(buzzer, 130); break;
      case 'D': setPwmFrequency(11, 256); analogWrite(buzzer, 140); break;
      case '#': setPwmFrequency(11, 256); analogWrite(buzzer, 150); break;
      case '*': setPwmFrequency(11, 1024); analogWrite(buzzer, 160); break;
      default : break;
    }
  }
}
```

```

    }
    delay(500);
    analogWrite(buzzer,0);
    Serial.println(customKey);
  }
}
/*
* Set pin 9's PWM frequency to 3906 Hz (31250/8 = 3906)
* Note that the base frequency for pins 3, 9, 10, and 11 is 31250 Hz
*setPwmFrequency(9, 8);

* Set pin 6's PWM frequency to 62500 Hz (62500/1 = 62500)
* Note that the base frequency for pins 5 and 6 is 62500 Hz
*setPwmFrequency(6, 1);
*
* The resulting frequency is equal to the base frequency divided by
* the given divisor:
*   - Base frequencies:
*       o The base frequency for pins 3, 9, 10, and 11 is 31250 Hz.
*       o The base frequency for pins 5 and 6 is 62500 Hz.
*   - Divisors:
*       o The divisors available on pins 5, 6, 9 and 10 are: 1, 8, 64,
*         256, and 1024.
*       o The divisors available on pins 3 and 11 are: 1, 8, 32, 64,
*         128, 256, and 1024.
*
* PWM frequencies are tied together in pairs of pins. If one in a
* pair is changed, the other is also changed to match:
*   - Pins 5 and 6 are paired on timer0
*   - Pins 9 and 10 are paired on timer1
*   - Pins 3 and 11 are paired on timer2
*/
void setPwmFrequency(int pin, int divisor) {
  byte mode;
  if(pin == 5 || pin == 6 || pin == 9 || pin == 10) {
    switch(divisor) {
      case 1: mode = 0x01; break;
      case 8: mode = 0x02; break;
      case 64: mode = 0x03; break;
      case 256: mode = 0x04; break;
      case 1024: mode = 0x05; break;
      default: return;
    }
    if(pin == 5 || pin == 6) {
      TCCR0B = TCCR0B & 0b11111000 | mode;
    } else {
      TCCR1B = TCCR1B & 0b11111000 | mode;
    }
  } else if(pin == 3 || pin == 11) {
    switch(divisor) {
      case 1: mode = 0x01; break;
      case 8: mode = 0x02; break;
      case 32: mode = 0x03; break;
      case 64: mode = 0x04; break;
      case 128: mode = 0x05; break;
      case 256: mode = 0x06; break;
    }
  }
}

```

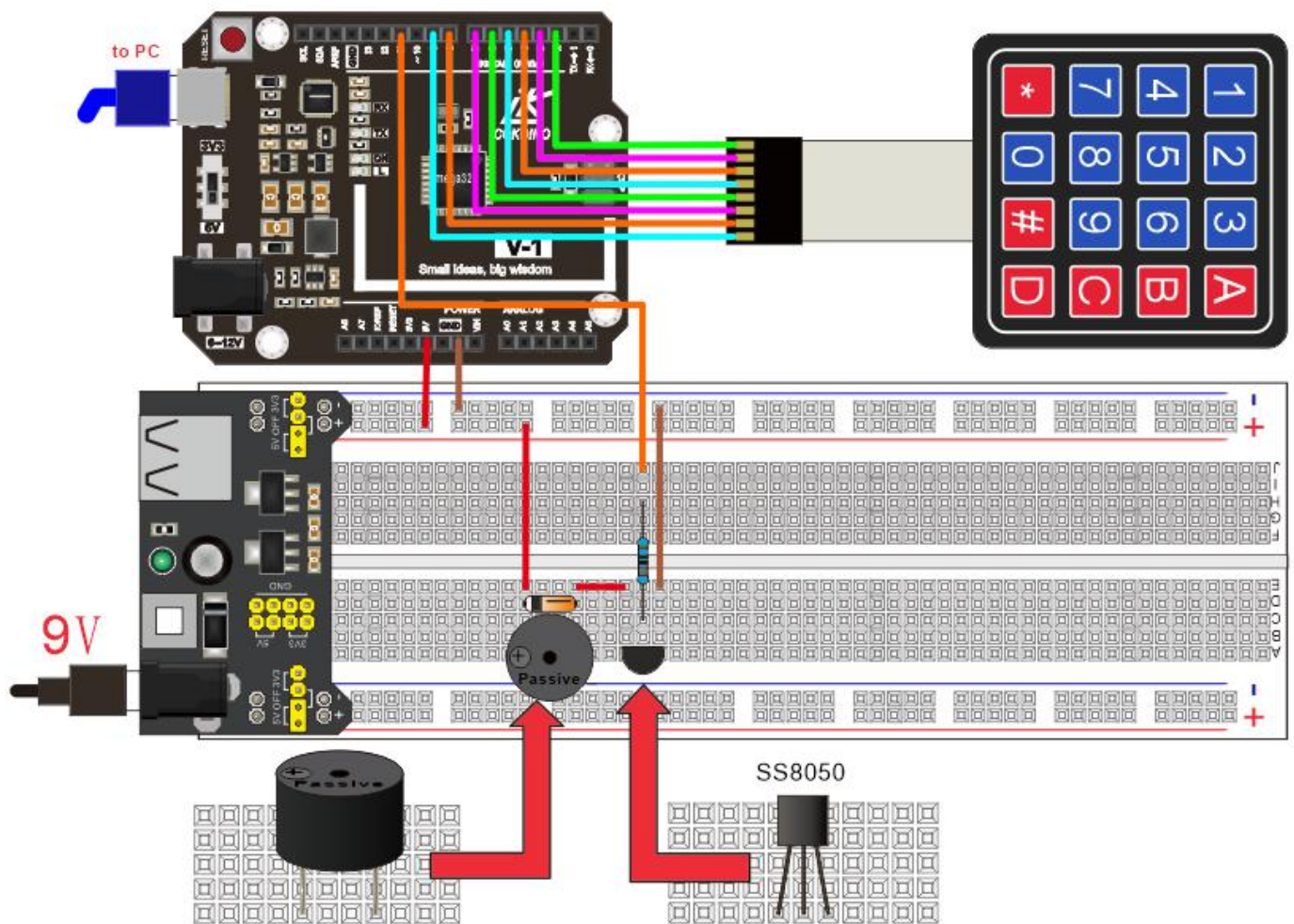


```

    case 1024: mode = 0x7; break;
    default: return;
  }
  TCCR2B = TCCR2B & 0b11111000 | mode;
}
}

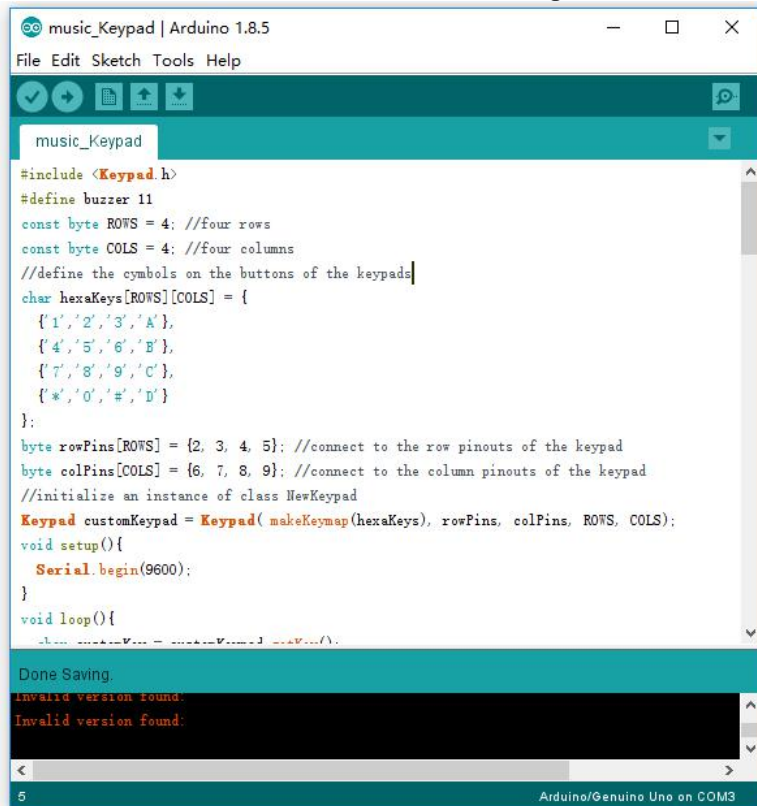
```

4.2、wiring diagram



4.3. Experimental procedure

4.3.1. Connect the UNO R3 motherboard to the COMPUTER via USB cable, and copy the above sample code into the Arduino IDE, as shown in the figure below:

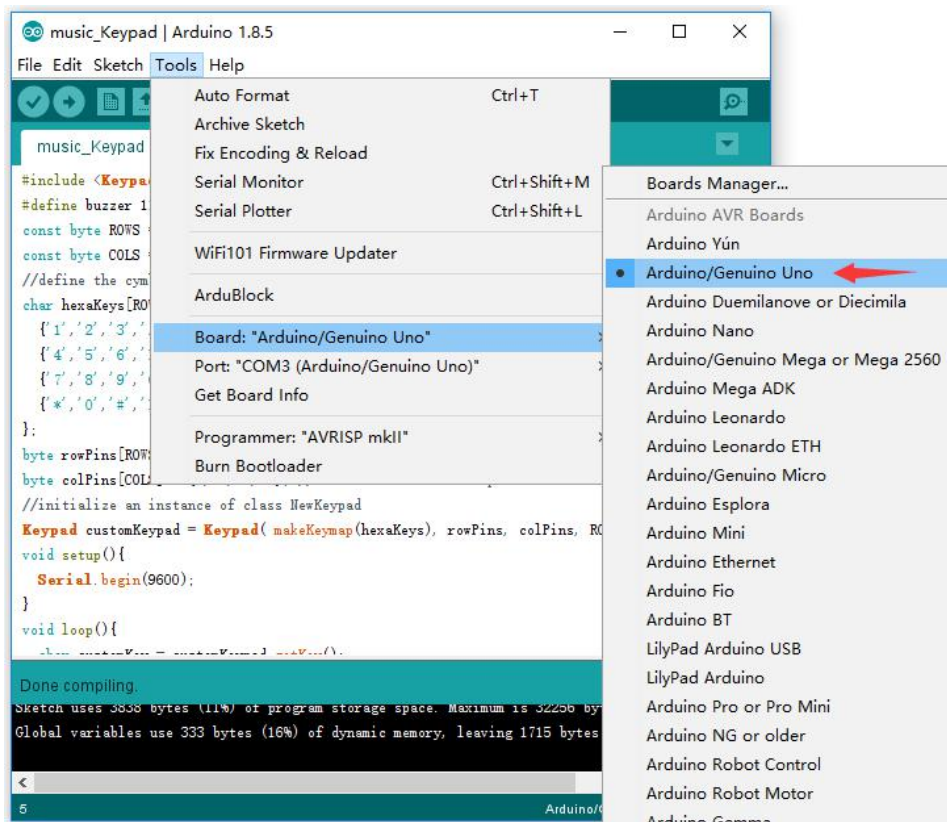


```
#include <Keypad.h>
#define buzzer 11
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the symbols on the buttons of the keypad
char hexaKeys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {2, 3, 4, 5}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {6, 7, 8, 9}; //connect to the column pinouts of the keypad
//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
void setup(){
  Serial.begin(9600);
}
void loop(){
  char customKey = customKeypad.getKey();
}
```

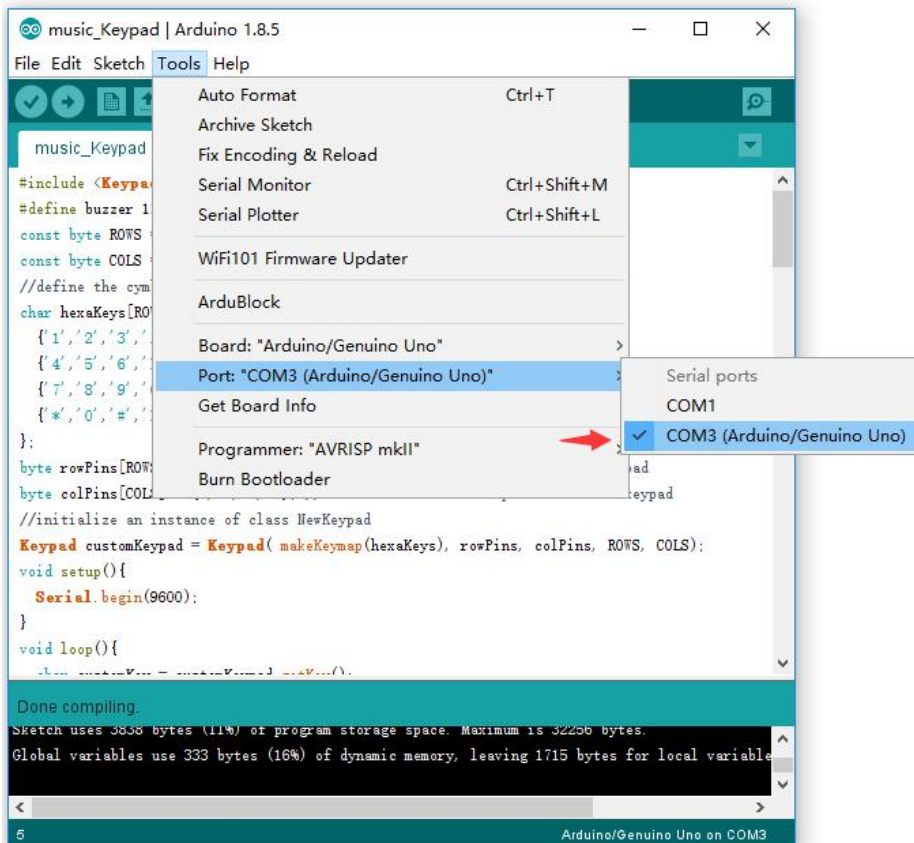
Done Saving.
Invalid version found.
Invalid version found.

5 Arduino/Genuino Uno on COM3

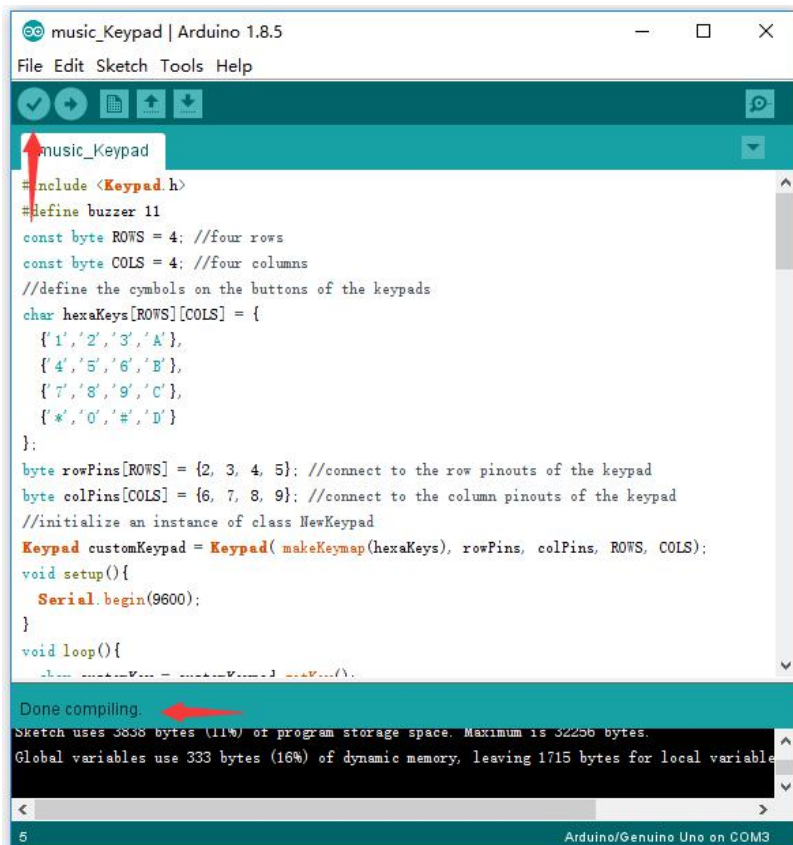
4.3.2. Select the main board type



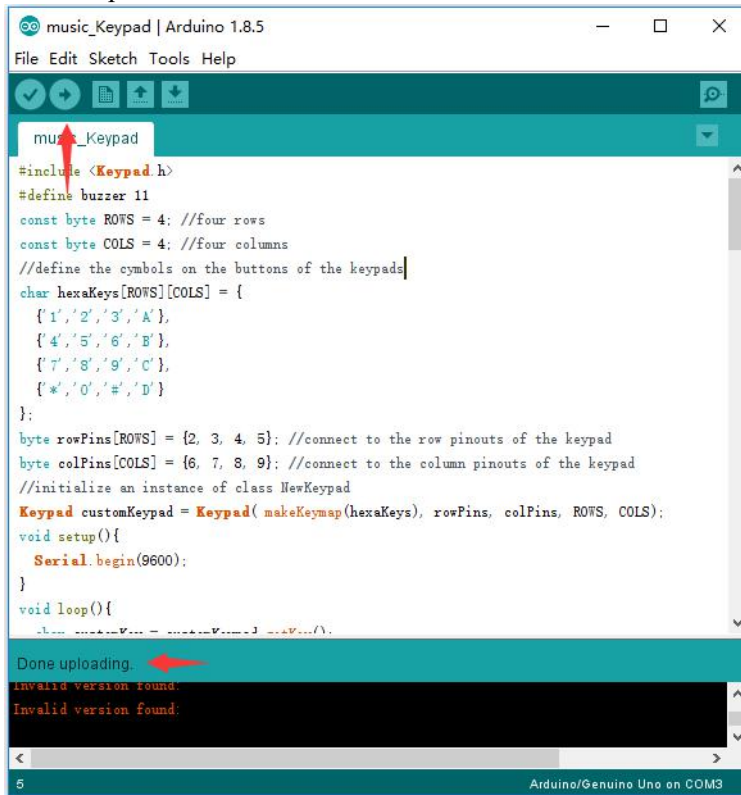
4.3.3、Choose a port



4.3.4、Check the code



4.3.5、Upload code



4.3.6、Plug in an external power source, turn on the IDE serial port monitor and adjust the baud rate to 9600. Press the keys on the 4*4 keyboard by hand, the buzzer will sound at different volume and frequency, and the serial port monitor will print the corresponding key value of the keypad, as shown in the figure below.

